Miami Isopycnic Coordinate Ocean Model (MICOM)

User's Manual

Details of the numerical code

G. Langlois, *Agence de Développements en Hydrodynamique et Océanographie Côtière, France*

Translated and revised from the original French by :

D. Brydon, Los Alamos National Laboratory, USA
R. Bleck, University of Miami, USA
S. Dean, Los Alamos National Laboratory, USA

Code version 2.6, Manual version 2.6A, February 7, 1997
This manual is available at: http://www.acl.lanl.gov/CHAMMP/micom.html
Report corrections to: micom_manual@acl.lanl.gov

Contents

Preface

0.1 Background

This manual was created in French by G. Langlois to describe MICOM, *code* version 2.5. This first English version of the manual, *manual* version 2.6A, is an English translation, with revisions that make it more compatible with MICOM *code* version 2.6. Note, however, that the code fragments in this manual are from *code* version 2.5.

0.2 Revisions to the Manual

Version 2.6A, February 7, 1997.

0.3 Acknowledgements

We thank Brice Rosenzweig, Los Alamos National Laboratory, for his help in understanding the original French.

0.4 Funding

This work has been supported by the United States' Department of Energy (DOE) Computer Hardware, Advanced Mathematics and Model Physics (CHAMMP) program. CHAMMP is one component of the DOE's Global Climate Change Program. MICOM is funded within the CHAMMP program to develop an efficient global isopycnic coordinate ocean model that is suitable for global climate studies.

Introduction

The University of Miami MICOM code is a primitive equation numerical model that describes the evolution of momentum, mass, heat and salt in the ocean. The theoretical support of the model has been described by BLECK *et al.* in a publication entitled "*Salinity-Driven Thermocline Transients in a Wind- and Thermohaline-Forced Isopycnic Coordinate Model of the North Atlantic*" (*JPO*, **22**, 1486-1505). Isopycnic ocean models use equations that have a coordinate of density in the vertical direction instead of the 'traditional' vertical coordinate of length. The difference between traditional ocean models - 'z coordinate' models - and isopycnic models, simply stated, is that the former predict water density changes at fixed depth levels, whereas the latter predict the depths at which certain density values are encountered. Thus, the traditional roles of *water density* and *height* as dependent and independent variables are reversed. To be specific, in MICOM, we use potential density (density corrected for compressibility effects) as the vertical model coordinate. After a change of variables over the vertical coordinate such that $\sigma : (x, y, z, t) \rightarrow (x, y, s(x, y, z, t), t))$, the general conservation equations written for the variable s $(= \rho)$ (BLECK 1978) are applied both in the isopycnic domain and in the surface mixed layer. The distribution of variables is organized according to the Arakawa C grid. The thermodynamic variables as well as the variables of motion are treated as layer variables. The isopycnic layers are indexed by their upper interfaces. The model uses a *split-explicit* numerical scheme to separately calculate the barotropic and baroclinic components of the field of prognostic variables (BLECK & SMITH 1990). The solution of the barotropic component is shifted in time. It is treated by a simple *forward-backward* scheme using the mass forward field in the continuity equation and the last pressure field in the equation of motion. The procedure to reproduce the flow of isopycnals is identical to that described in BLECK & BOUDRA (1986). Each isopycnic layer is able to alternatively appear or disappear in each point of the domain. The treatment of the problem of the intersection of isopycnals with the bathymetry is analogous to that worked out in BLECK & SMITH (1990). The forcing of the model is affected by two functions of turbulent heat flux and momentum. The effect of the wind is confined to the mixed layer whose thickness can not exceed that of the Ekman layer. The interaction modes of the mixed layer with the subadjacent isopycnic layers are described in BLECK *et al.* (1989). The treatment of the equations of horizontal advection-diffusion of heat and salt is accomplished by a third-order conservation scheme derived from the work of SMOLARKIEWICZ & CLARK (1986) and SMOLARKIEWICZ & GRABOWSKI (1990). The model accommodates a user specified, horizontal geographic zone. An example domain extends from 10^o to 60^oN with a spatial resolution of 1^o. The vertical stratification is reproduced by choosing the number of isopycnic layers, for example 10, and specifying their respective densities.

1 FORTRAN 77 Version 2.6

In version 2.6 of the FORTRAN code of MICOM, the *main* program is named `micom.f`. The dimensions of various variables are introduced by means of symbolic constants. The corresponding `parameter` statements are grouped in the file **dimensions.h** (*cf.* § 1.1.1). The dimension declarations are grouped in the file **common_blocks.h** (*cf.* § 1.1). The program makes calls to different statement functions grouped in the file **stmt_functions.h** (*cf.* § 13).

Before iterative integration calls, it is necessary to define the different characteristics of a simulation. One can summarize the setup of MICOM 2.6 in the following manner :

I. The first step consists of introducing the bathymetry (*cf.* § 1.4.4) and to take steps to distinguish the submerged zones from those on land. (*cf.* § 1.4.4).

II. The second phase has the goal of characterizing the projection used. Several options may be used. Generally, the system of equations is solved on a Mercator grid with the x axis indicating South and the y axis pointing East. It is possible to use a Mercator grid whose poles are rotated to the equator by changing the logical variable `rotat` (*cf.* § 1.4.2). With this choice, one needs to calculate the associated spatial parameters: mesh size, Coriolis parameter, *etc.*

III. Initialization of the variables is done in the third step (*cf.* § 1.2).

IV. The iterative computations are based on 9 main procedures. Each of these subroutines is listed in the order called along with the function it performs :

`subroutine cnuity` : continuity equation ;
`subroutine tsadvc` : advection equation ;
`subroutine momeq1` : momentum equations ;
`subroutine momeq2` : momentum equations ;
`subroutine barotp` : dynamic barotropic mode ;
`subroutine convec` : vertical convection ;
`subroutine diapfl` : diapycnal mixing ;
`subroutine thermf` : ocean-atmosphere exchanges ;
`subroutine mxlayr` : evolution equation of the surface mixed layer.

MICOM's scheme needs results of the previous two computations. These results are saved via a third dimension. The index `km` represents the solution at time step $n\Delta t$ and the index `kn` represents alternately the results at $(n-1)\Delta t$ and $(n+1)\Delta t$.

V. After each iteration, the version 2.6 of MICOM offers a certain number of outputs (writing files, graphical output, *etc.*) as well as 2 control tests. (*cf.* § 1.3.2).

1.1 Declarations

1.1.1 Symbolic Constants

The different parametric constants have been grouped in the following list :

`idm,ii,ii1` number of computational rows

`jdm,jj,jj1`	number of computational columns
`kdm,kk`	number of computational layers
`ms,msd`	maximum number of submerged segments in rows or columns
`nlato,nlongo`	grid dimensions in latitude/longitude (i,j) direction
`nlatn,nlongn`	grid dimensions in (rotated) (i,j) direction
`athird`	1/3

The number of layers is fixed by the statement : `parameter (kdm=...)`

1.1.2 Variables of state and auxiliary variables

The **common_blocks.h** file contains an unlabeled `common` that declares the following variables :

`u(idm,jdm,2*kdm),uold(idm,jdm,kdm)`	u component of velocity
`v(idm,jdm,2*kdm),vold(idm,jdm,kdm)`	v component of velocity
`dp(idm,jdm,2*kdm),dpold(idm,jdm,kdm)`	layer thickness
`dpu(idm,jdm,2*kdm),dpv(idm,jdm,2*kdm)`	layer thickness at the computational point of variables u and v
`p(idm,jdm,kdm+1)`	interface pressure
`pu(idm,jdm,kdm+1),pv(idm,jdm,kdm+1)`	interface pressure at the computational point of variables u and v
`corio(idm,jdm)`	Coriolis parameter
`psikk(idm,jdm)`	Montgomery potential in the bottom layer
`vort(idm,jdm),potvor(idm,jdm)`	vorticity and potential vorticity
`thmix(idm,jdm,2)`	mixed layer density
`temp(idm,jdm,2*kdm)`	temperature
`saln(idm,jdm,2*kdm)`	salinity
`sdot(idm,jdm)`	mixed layer entrainment velocity (generalized vertical velocity)
`montg(idm,jdm,kdm)`	Montgomery potential
`defor1(idm,jdm),defor2(idm,jdm)`	deformation components
`ubavg(idm,jdm,3),vbavg(idm,jdm,3)`	barotropic velocity
`pbavg(idm,jdm,3)`	barotropic pressure

`ubrhs(idm,jdm),vbrhs(idm,jdm)`	baroclinic forcing terms (right hand side) in barotropic u,v equations
`utotm(idm,jdm),vtotm(idm,jdm)`	u, v components of total (barotropic+baroclinic) velocity
`utotn(idm,jdm),vtotn(idm,jdm)`	u, v components of total (barotropic+baroclinic) velocity at 2 time levels
`uflux(idm,jdm),vflux(idm,jdm)`	horizontal mass fluxes
`uflux1(idm,jdm),vflux1(idm,jdm)`	” ” ”
`uflux2(idm,jdm),vflux2(idm,jdm)`	” ” ”
`uflux3(idm,jdm),vflux3(idm,jdm)`	” ” ”
`uflx(idm,jdm,kdm),vflx(idm,jdm,kdm)`	” ” ”
`util1(idm,jdm),util2(idm,jdm)`	temporary storage arrays
`util3(idm,jdm),util4(idm,jdm)`	” ” ”
`gradx(idm,jdm),grady(idm,jdm)`	u, v components of horizontal pressure gradient
`pgfx(idm,jdm),pgfy(idm,jdm)`	u, v components of horizontal pressure force
`scu(idm),scv(idm),scp(idm)`	grid scale at the u,v,p points
`scu2(idm),scv2(idm),scp2(idm)`	grid scales squared
`scui(idm),scvi(idm),scpi(idm)`	inverses of `scu,scv,scp`
`scu2i(idm),scv2i(idm),scp2i(idm)`	inverses of `scu2,scv2,scp2`
`depthu(idm,jdm),depthv(idm,jdm)`	bottom pressure at u,v points
`pvtrop(idm,jdm)`	potential vorticity of barotropic flow
`depths(idm,jdm)`	water depth
`drag(idm,jdm)`	bottom drag
`visc(idm,jdm)`	eddy viscosity
`uja(idm,jdm),ujb(idm,jdm)`	velocities at lateral u neighbor points
`via(idm,jdm),vib(idm,jdm)`	” ” ” v ” ”
`wgtia(idm,jdm),wgtib(idm,jdm)`	submerged sidewall weights, indicate presence of land at neighbor point
`wgtja(idm,jdm),wgtjb(idm,jdm)`	” ” ” ” ” ”
`pbot(idm,jdm)`	bottom pressure at t=0

`tracer(idm,jdm,kdm)`	inert tracer concentration (optional)
`thup(idm,kdm),thdn(idm,kdm)`	bulk theta above and below the interface of row `k`
`sgain(idm,kdm)`	variation of salinity from diapyc. mix.
`surflx(idm,jdm)`	thermal energy flux through the surface
`buoyfl(idm,jdm)`	surface buoyancy flux
`ustar(idm,jdm)`	friction velocity
`turgen(idm,jdm)`	turbulent kinetic energy generation
`dpmx(idm,jdm)`	maximum thickness at neighbor points
`tdp(idm,jdm),sdp(idm,jdm)`	vertical heat and salt integrals
`klist(idm,jdm)`	index of the layer under the mixed layer

1.1.3 Logical variables

`common/swtchs/` contains the logical variables :

`diagno`	enables diagnostic messages
`rotat`	90 degree rotation of the poles of the Mercator projection
`thermo`	enable thermodynamic forcing functions
`windf`	include wind stress in forcing functions
`vthenu`	used in *forward-backward* differences scheme
`trcrin`	read initial tracer field from restart file
`trcout`	advect tracer field and save it in restart file

1.1.4 Atmospheric forcings

The monthly average values of atmospheric forcings are contained in `common/frcing/` :

`taux(idm,jdm,12)`	wind stress in x direction
`tauy(idm,jdm,12)`	wind stress in y direction
`wndspd(idm,jdm,12)`	wind speed (tke source)
`radflx(idm,jdm,12)`	net short-wave radiation flux
`airtmp(idm,jdm,12)`	pseudo air temperature
`precip(idm,jdm,12)`	precipitation
`vapmix(idm,jdm,12)`	atmospheric water vapor mixing ratio

`rmu(nbdy)`	weights for lateral boundary condition relaxation
`pwall(2*nbdy,jdm,kdm,12)`	pressure boundary condition along sidewalls
`swall(2*nbdy,jdm,kdm, 4)`	salinity boundary condition along sidewalls
`twall(2*nbdy,jdm,12)`	mixed layer temperature boundary condition along sidewalls

1.1.5 Numerical parameters

`common/varbls/` collects various numerical parameters :

```
common/varbls/nstep,nstep1,nstep2,time,time0,delt1,lstep,dlt,l0,l1,l2,l3,
common/varbls/time,delt1,dlt,w0,w1,w2,w3,ws0,ws1,ws2,ws3,
area,watcum,empcum,nstep,nstep1,nstep2,lstep,l0,l1,l2,l3,ls0,ls1,ls2,ls3
```

where :

`delt1`	baroclinic time step
`l0,l1,l2,l3`	indices of monthly atmospheric forcing files
`w0,w1,w2,w3`	time interpolation coefficients of atmospheric forcing

`common/parms1/` contains the following parameters :

`theta(kdm)`	used to calculate specific volume in layer `k`: $alpha = 1/rho = thref \times (1 - theta(k))$
`thbase`	base value of specific volume
`baclin`	baroclinic time step
`batrop`	barotropic time step
`thkdff`	diffusion velocity (cm/s) for thickness diffusion (diffusivity divided by mesh size gives cm/s)
`veldff`	diffusion velocity (cm/s) for momentum dissipation
`temdff`	diffusion velocity (cm/s) for temperature
`viscos`	coefficient of nonlinear viscosity
`diapyc`	diapycnal diffusivity times buoyancy frequency (cm^2/s^2)
`vertmx`	diffusion velocity (cm/s) for momentum mixing at the mixed layer base
`mixfrq`	number of time steps between diapycnal mixing calculations
`h1`	depth interval used in lateral weighting of horizontal pressure gradient

`slip`	slip = +1 for free-slip boundary condition, slip = -1 for non-slip condition
`cbar`	rms flow speed (cm/s) for linear bottom friction law
`diagfq`	number of days between model diagnostics
`ntracr`	number of time steps between tracer transport
`wuv1, wuv2`	weights for time smoothing of u, v field
`wts1, wts2`	weights for time smoothing of t, s field
`wbaro`	weight for time smoothing of barotropic u, v, p field
`thkmin`	minimum mixed-layer thickness (m)
`thkbot`	thickness of bottom boundary layer (pressure units)

1.1.6 Constants

The constants are grouped in `common/consts/` :

`tenm`	pressure thickness value corresponding to 10 m of water thickness
`onem`	" " 1 m " "
`tencm`	" " 10 cm " "
`onecm`	" " 1 cm " "
`onemm`	" " 1 mm " "
`g`	gravitational acceleration
`csubp`	specific heat of air at constant pressure ($J/g/\ ^oC$)
`spcifh`	specific heat of sea water ($J/g/\ ^oC$)
`cd`	drag coefficient
`ct`	thermal transfer coefficient
`airdns`	air density at sea level (g/cm^3)
`evaplh`	latent heat of evaporation (J/g)
`thref`	reference value of specific volume (cm^3/g)
`epsil`	small nonzero number used to prevent division by zero

1.1.7 Assignments

All the parameters are initialized by a *block data* sub-program whose statements are in **blkdat.f**. For example, the specific volume of each layer is initialized by the statement :

```
c
c --- specific volume in layer k is alpha = 1/rho = thref x ( 1 - theta(k) )
      data theta/.02500,.02550,.02595,.02635,.02670,
     .           .02700,.02725,.02745,.02760,.02770/
c
```

Note

MICOM 2.6 uses the CGS system of units. For example the gravitational constant is specified by the statement :

```
c
c --- 'g' = gravitational acceleration
      data g/980.6/
c
```

To know the relation between the difference of pressure existing between the upper and lower limits of a slice of water of unit density and the distance between these interfaces, we introduce the constants :

```
c
c --- layer thicknesses in units of pressure:
      data tenm,onem,tencm,onecm,onemm/980600.,98060.,9806.,980.6,98.06/
c
```

1.2 Initializations

The field variables are initialized by the two successive steps :

1) Call the *subroutine* **inicon.f** whose main function is to set all initial values to zero. It is also during this step that we calculate the Montgomery potential (*cf.* § 4) and the potential vorticity of the model ocean at rest.

2) Reading of data from the preceding run.

1.3 Running MICOM

1.3.1 Makefile

All the elements necessary to create an executable MICOM (compilation, etc.) are specified in a makefile. The different source files are referenced in the macro `SCRS` within the makefile :

```
#
```

```
SRCS = micom.f advem.f barotp.f bigrid.f blkdat.f cnuity.f convec.f \
       diapfl.f dpudpv.f forfun.f indxi.f indxj.f inicon.f momeq1.f \
       momeq2.f mxlayr.f oldnew.f overtn.f pakk.f pakmsk.f poflat.f \
       prtmsk.f psmoo.f restart.f thermf.f tsadvc.f zebra.f
#
```

Similarly, the different object files are referenced in the macro `OBJS` :

```
#
OBJS = micom.o advem.o barotp.o bigrid.o blkdat.o cnuity.o convec.o \
       diapfl.o dpudpv.o forfun.o indxi.o indxj.o inicon.o momeq1.o \
       momeq2.o mxlayr.o oldnew.o overtn.o pakk.o pakmsk.o poflat.o \
       prtmsk.o psmoo.o restart.o thermf.o tsadvc.o zebra.o
#
```

The MICOM makefile specifies the loading or linking by :

```
#
micom : $(OBJS)
        f77 -v -o micom $(OBJS)
#
```

Similarly the generation of each object file is specified by :

```
#
mxlayr.o : mxlayr.f dimensions.h common_blocks.h stmt_functions.h
          f77 $(FFLAGS) -c mxlayr.f
#
```

The macro `FLAGS` represents options for the `f77` compiler :

```
#
FLAGS = -O
#
```

In R. Bleck's notes (*cf.* § 14), he indicates that on some machines it is more efficient to use powers of 2 to dimension the symbolic constants `idm` and `jdm`, while on others it is less.

1.3.2 Output for testing purposes

The coordinates of the point where detailed results are printed are specified in `common/testpt/` :

```
common/testpt/itest,jtest
```

Version 2.6 of MICOM does two types of tests :

1) Calculation of the barotropic stream function. This is done with an overrelaxation method in the subroutine **poisnd.f.**

2) Estimation of the meridional overturning rate. This is done in the subroutine **overtn.f**, where we calculate the zonally averaged meridional heat flux and display it in stream function form.

1.4 Configuring MICOM

1.4.1 Implementation

The geometry of the application domain is stored in the file **dimensions.h**. The horizontal extent is defined by the 2 parameters `idm` and `jdm`.
The specification of islands and continents is done with the help of the parameter `ms`. It sets the maximum number of interruptions of the oceanic domain in rows or in columns plus one. The equivalent in the diagonal direction is represented by the parameter `msd`. For each of the 4 points of the C grid $(u, v, \Delta p, Q)$, `common/gindex/` contains seven tables of entries.

1) one table of dimension `idm*jdm` : `iu(idm,jdm)`. At submerged points, `iu(i,j)=1`; else, `iu(i,j)=0`.
2) one table comprising the number of submerged segments in the row i : `isu(jdm)`;
3) a table giving the lower limits of each segment by column : `ifu(jdm,ms)` ;
4) a table giving the upper limits of each segment by column : `ilu(jdm,ms)` ;
5) a table comprising the number of segments of the column j : `jsu(idm)` ;
6) a table of lower limits of each segment by row : `jfu(idm,ms)` ;
7) a table of upper limits of each segment by row : `jlu(idm,ms)`.

A segment is defined as a number of contiguous submerged points. To represent the separation between land and sea along the diagonals, `common/diags/` contains 5 tables whose values come from numerical processing of the bathymetry file (*cf.* § 1.4.4):)

1) a table of the number of segments in each diagonal : `nsec(idm+jdm)`;
2) a table of the abscissas of the lower limits of each segment : `ifd(idm+jdm,msd)` ;
3) a table of the abscissas of the upper limits of each segment : `ild(idm+jdm,msd)` ;
4)) a table of the ordinates of the lower limits of each segment : `jfd(idm+jdm,msd)` ;
5) a table of the ordinates of the upper limits of each segment : `jld(idm+jdm,msd)`.

This information is only used in diagnosing the barotropic stream function.

1.4.2 Projections

Two projections can be used. The projection is chosen by setting the variable `rotat`. The different assignments used are in `common/pivot/` :

```
common/pivot/xpivo,ypivo,gridо,xpivn,ypivn,gridn
```

1) `rotat=.false.` : normal Mercator projection (x pointing South and y pointing East). `xpivn` specifies the i index of the equator (`xpivo,ypivo,ypivn` are not used) ;

2) `rotat=.true.` : Mercator projection after a 90 degree rotation (x pointing East and y pointing North). In this case, `xpivo,ypivo` indicate respectively the coordinates i, j of the pivot in the standard grid. In the grid after the rotation, `xpivo,ypivo` and `xpivn,ypivn` reference the coordinates of two points situated on the equator. (figure 1).

Typically, we initialize the variables by the statement :

```
      data xpivn,gridn/41.,2./
```

`gridn` is the mesh size of the latitude/longitude grid in degrees and `gridn` is the mesh size of the actual model grid (whether rotated by $\pi/2$ or not) in degrees longitude.
The subroutine **newold.f** computes the coordinates of points in the rotated grid as a function of their positions in the original grid. The subroutine **oldnew.f** carries out the inverse function.

1.4.3 Mesh

The size of the mesh is a function of latitude which is itself a function of the distance between a point and the equator. The two functions giving respectively the latitude from the grid distance and its inverse respectively are found in the file **stmt_functions.h** :

```
c
c --- formulae relating latitude to grid distance from equator
c
      alat(dist1,grid)=(2.*atan(exp(dist1*grid/radian))-pi/2.)
      dist(alat1,grid)=alog(tan((2.*alat1+pi)/4.))*radian/grid
c
```

1.4.4 Bathymetry

The distribution of variables in a C grid is such that only `idm-1*jdm-1` depths are required to represent the bathymetry of the 'rectangular' portion of the ocean considered. (*cf.* § 11). By convention, land zones are given a depth of zero. The distinction between exposed and submerged zones is carried out by the call :

```
c
c --- determine do-loop limits for u,v,p,q points
      call bigrid(depths)
c
```

The boundaries of the segments along diagonals are needed by the optional Poisson solver (*cf.* § 1.3.2).) To integrate the system of equations over the whole domain, the boundaries of segments by rows and by columns have to be determined for the four points which bound the mesh (*cf.* § 11). To do this, in **bigrid.f**, we have the statements :

```
c
c --- determine loop indices for mass and velocity points
      call indxi(ip,ifp,ilp,isp)
      call indxj(ip,jfp,jlp,jsp)
      call indxi(iu,ifu,ilu,isu)
      call indxj(iu,jfu,jlu,jsu)
      call indxi(iv,ifv,ilv,isv)
      call indxj(iv,jfv,jlv,jsv)
```

```
c
c --- determine loop bounds for vorticity points, including interior and
c --- promontory points
      call indxi(iq,ifq,ilq,isq)
      call indxj(iq,jfq,jlq,jsq)
c
```

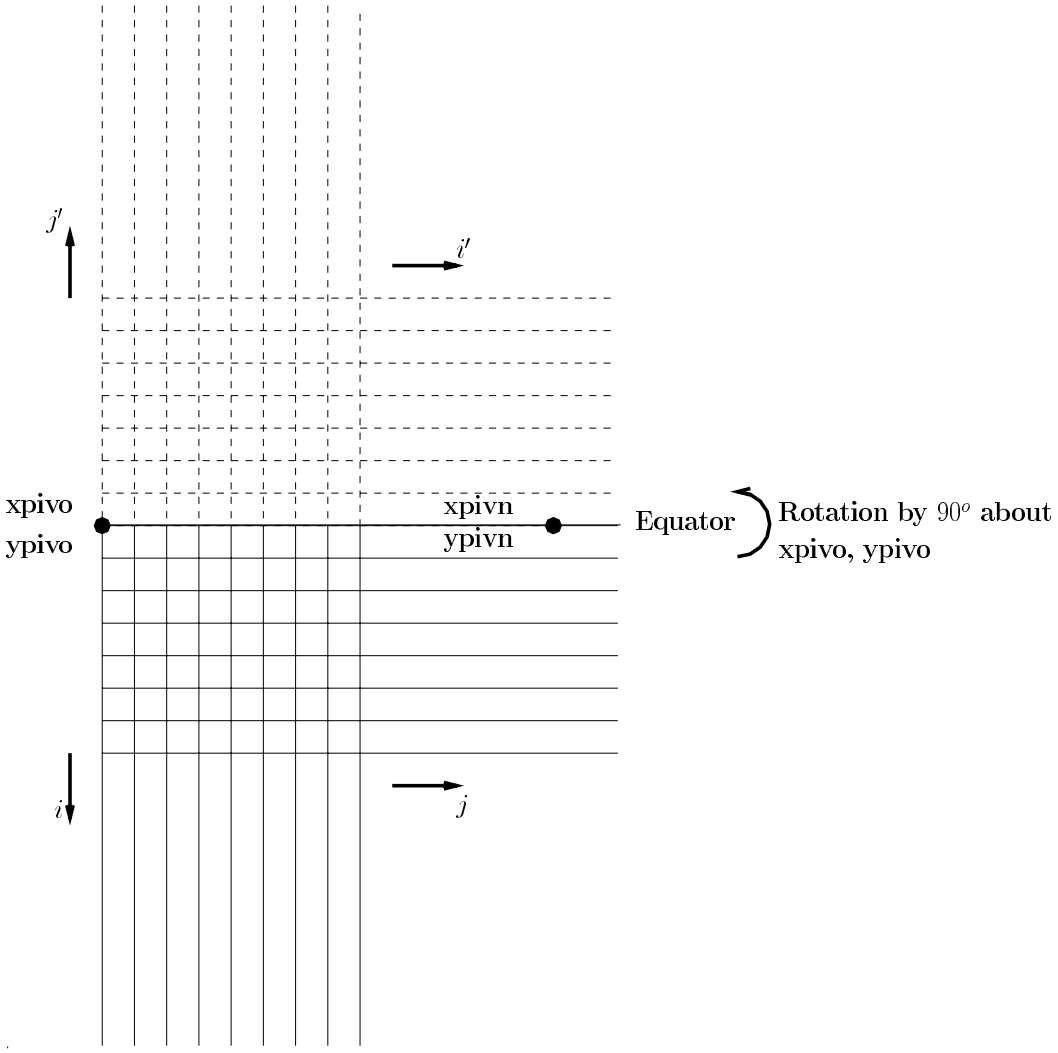

Figure 1: *Rotation of the initial grid and positions of segments representing the equator in each of 2 grids.*

2 Continuity equation : cnuity.f

2.1 Formalism and numerical techniques

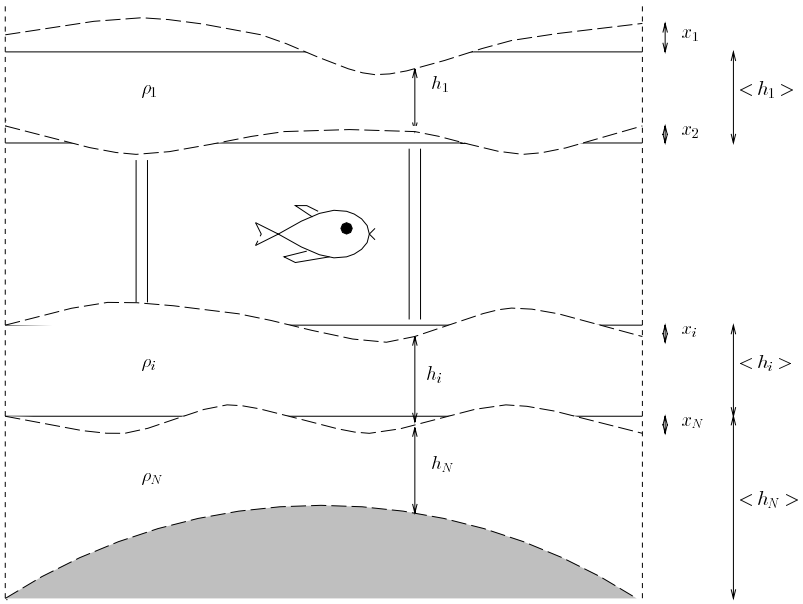

Figure 2: *Vertical discretization of the multilayer ocean*

BLECK & SMITH (1990) assume that the difference in pressure between the two interfaces of the k^{th} layer has the form :

$$\Delta p_k = (1+\eta)\Delta p'_k \tag{1}$$

with $\Delta p_k = g\rho_k h_k$ and $\Delta p'_k = g\rho_k h'_k$. Elsewhere, we introduce the decomposition :

$$\mathbf{u}_k = \overline{\mathbf{u}} + \mathbf{u}'_k \tag{2}$$

with :

$$\overline{\mathbf{u}} = \frac{\sum_{k=1}^{N} \rho_k h_k \mathbf{u}_k}{\sum_{k=1}^{N} \rho_k h_k} \tag{3}$$

and :

$$\overline{\mathbf{u}'_k} = 0 \tag{4}$$

The tendency equation for the component $\Delta p'$ entering in the total expression for the change of pressure in the layer k is then written (BLECK & SMITH, 1990) :

$$\frac{\partial}{\partial t}\Delta p'_k + \nabla \cdot (\mathbf{u}\Delta p')_k = \frac{\Delta p'_k}{p'_b}\nabla \cdot (\overline{\mathbf{u}} p'_b) \tag{5}$$

Under the hydrostatic hypothesis, with a free surface, we have :

$$p'_b = \sum_{k=1}^{N} \Delta p'_k = g \sum_{k=1}^{N} \rho_k h'_k = g \rho_r H \tag{6}$$

with H the water depth :

$$H(x,y) = \sum_{k=1}^{N} \langle h_k \rangle = \sum_{k=1}^{N} h_k - \xi_1 \tag{7}$$

$\langle h_k \rangle$ represents the initial thickness of the layer k and ξ_1 represents the change in the free surface (*c.f.* figure 2). ρ_r is the column mean density.

2.1.1 FCT (Flux-Corrected Transport) scheme

The base FCT scheme from the work of ZALESAK (1986) is outlined here in 7 steps :

1) The first inference of the variable $\Delta p'_k$ is made by introducing a classic upstream scheme. We denote it by $\Delta p'^{up}_{i,j,k}$ in the layer k. The diffusive fluxes of this first step are calculated from the total velocity field **u**. Consider the problem (5) without its second term in a one-dimensional form and take $\mathbf{P'}_k = (\mathbf{u}\Delta p')_k$. Then we can write :

$$P'^{up}_{i+1/2,k} = (u\Delta p')^{up}_{i+1/2,k} = \begin{cases} u^{mid}_{i+1/2,k}\ \Delta p'^{old}_{i,k} & \text{if } u_{i+1/2,k} > 0 \\ \\ u^{mid}_{i+1/2,k}\ \Delta p'^{old}_{i+1,k} & \text{if } u_{i+1/2,k} < 0 \end{cases} \tag{8}$$

where *mid* and *old* refer to two successive instants in the leapfrog scheme. In a donor-cell scheme, the upstream fluxes as well as the velocity should be defined at the interfaces between the cells.

2) Always in using the total velocity field, we proceed next to calculate the non-diffusive flux by a scheme second-order in space and centered in time :

$$P'^{*}_{i+1/2,k} = (u\Delta p')^{*}_{i+1/2,k} = u^{mid}_{i+1/2,k}\ \frac{\Delta p'^{mid}_{i,k} + \Delta p'^{mid}_{i+1,k}}{2}. \tag{9}$$

3) We introduce the flux of anti-diffusion **A** such that : $A_{i+1/2,k} = P'^{*}_{i+1/2,k} - P'^{up}_{i+1/2,k}$. To assure the stability of the scheme (*i.e.* to counter the appearance of negative values of $\Delta p'^{new}_{i,k}$), we substitute for the anti-diffusive flux **A** with the corrected flux $\mathbf{A}^c$ such that : $A^c_{i+1/2,k} = C_{i+1/2,k} A_{i+1/2,k}$ and with : $0 \leq C \leq 1$. For $C = 0$, we restore the flux of order 1 and $C = 1$ gives back the flux of order 2. The final solution is expressed by a combination of fluxes of orders one and two which moderates the perturbations of stability which appear. For each layer, we have (BARAILLE & FILATOFF, 1995) :

$$C_{i+1/2,k} = \begin{cases} \min\left(R^{+}_{i+1,k}\tilde{;}\ R^{-}_{i,k}\right) & \text{if } A_{i+1/2,k} \geq 0 \\ \\ \min\left(R^{+}_{i,k}\tilde{;}\ R^{-}_{i+1,k}\right) & \text{if } A_{i+1/2,k} < 0 \end{cases} \tag{10}$$

where we successively introduce :

$$\mathbf{I} \begin{cases} P^+_{i,k} = \max\left(0, A_{i-1/2,k}\right) - \min\left(0, A_{i+1/2,k}\right) \\ P^-_{i,k} = \max\left(0, A_{i+1/2,k}\right) - \min\left(0, A_{i-1/2,k}\right) \end{cases} \tag{11}$$

then :

$$\mathbf{II} \begin{cases} Q^+_{i,k} = \Delta p'^{max}_{i,k} - \Delta p'_{i,k} \\ Q^-_{i,k} = \Delta p'_{i,k} - \Delta p'^{min}_{i,k} \end{cases} \tag{12}$$

and

$$\mathbf{III} \begin{cases} R^+_{i,k} = \begin{cases} \min\left(1, \frac{\Delta x}{\Delta t}\frac{Q^+_{i,k}}{P^+_{i,k}}\right) & \text{if } P^+_{i,k} > 0 \\ 0 \text{ if } P^+_{i,k} = 0 \end{cases} \\ R^-_{i,k} = \begin{cases} \min\left(1, \frac{\Delta x}{\Delta t}\frac{Q^-_{i,k}}{P^-_{i,k}}\right) & \text{if } P^-_{i,k} > 0 \\ 0 \text{ if } P^-_{i,k} = 0 \end{cases} \end{cases} \tag{13}$$

$R^+_{i,k}$ and $R^-_{i,k}$ represent the biggest multiplicative factors of anti-diffusive flux which assure respectively : $\Delta p'^{n+1}_{i,k} \leq \Delta p'^{max}_{i,k}$ and $\Delta p'^{n+1}_{i,k} \geq \Delta p'^{min}_{i,k}$ with :

$$\begin{cases} \Delta p'^{max}_{i,k} = \max(\Delta p'^{n}_{i-1,k}, \Delta p'^{n}_{i,k}, \Delta p'^{n}_{i+1,k}) \\ \Delta p'^{min}_{i,k} = \min(\Delta p'^{n}_{i-1,k}, \Delta p'^{n}_{i,k}, \Delta p'^{n}_{i+1,k}) \end{cases} \tag{14}$$

The formulas corresponding to the bidimensional case are given in BARAILLE & FILATOFF (1995).

4) In summing the form (5) without the second term over the N layers of the vertical discretization, as we have $\partial p'_b/\partial t = 0$, we obtain :

$$\mathbf{P'} = \sum_{k=1}^{N} \nabla \cdot \mathbf{P'}_k = 0 \tag{15}$$

It is clear that to satisfy this condition, a second order approximation of the flux $\mathbf{P'}$ is better than any lower order approximation. We then write the identity :

$$\sum_{k=1}^{N} \left(P'^{*}_{i+1/2,k} - P'^{*}_{i-1/2,k}\right) = 0 \tag{16}$$

The sum of anti-diffusive flux corrections over the vertical introduces a bias in the conservation of p'_b for which we must compensate. To do this, we calculate the vertical sum of flux corrections (by means of a second order approximation) :

$$\mathcal{A}_{i+1/2} = \sum_{k=1}^{N} \left(1 - C_{i+1/2,k}\right) \mathbf{A}_{i+1/2,k} \tag{17}$$

5) To evaluate the effect of the anti-diffusive flux "integral", we calculate by layer the new thickness :

$$\Delta p'^{\dagger}_{i,k} = \Delta p'^{up}_{i,k} - \Delta t \ (\nabla \cdot \mathbf{A}^c)_{i,k} \tag{18}$$

From that we obtain the bottom pressure :

$$p'^{\dagger}_{b_i} = \sum_{k=1}^{N} \Delta p'^{\dagger}_{k} \tag{19}$$

6) Unless we have $C_{i+1/2,k} = 1$ for each layer, $p'^{\dagger}_{b_i} \neq p'_{b_i}$. To remedy this problem, we make a second correction B to the upstream flux such that :

$$B_{i+1/2,k} = \frac{\Delta p'^{\dagger}_{i,k}}{p'^{\dagger}_{b_i}} \mathcal{A}_{i+1/2} \tag{20}$$

The final flux takes the form :

$$P'^{fin}_{i+1/2,k} = P'^{up}_{i+1/2,k} + A^c_{i+1/2,k} + B_{i+1/2,k} \tag{21}$$

and we rectify the preceding estimation $\Delta p'^{\dagger}_{i,k}$ by the relation :

$$\Delta p'^{\ddagger}_{i,k} = \Delta p'^{\dagger,k}_{i,k} - \Delta t \ (\nabla \cdot \mathbf{B})_{i,k} \tag{22}$$

where we put the bottom pressure to :

$$p'^{\ddagger}_{b_i} = \sum_{k=1}^{N} \Delta p'^{\ddagger}_{i,k} \tag{23}$$

7) The last step is to account for the right hand side of the original equation (5). Once we are at this stage, we will claim that the flux was corrected to best satisfy $\partial p'_b / \partial t = 0 \tilde{;} \ \forall i$. From that, we can therefore write for each layer k :

$$\frac{\partial}{\partial t} \Delta p'^{fin}_{k} + \nabla \cdot \mathbf{P}'^{fin}_{k} = \frac{\Delta p'^{fin}_{k}}{p'_b} \nabla \cdot (\overline{\mathbf{u}} p'_b) \tag{24}$$

as we, of course, are looking to eventually satisfy :

$$\sum_{k=1}^{N} \Delta p'^{fin}_{k} = p'_b. \tag{25}$$

For all points, the form (24) is summed over the vertical giving that :

$$\nabla \cdot \left(\sum_{k=1}^{N} \mathbf{P}'^{fin}_{k} \right) = \nabla \cdot (\overline{\mathbf{u}} p'_b). \tag{26}$$

In the divergence term, the vertical flux average and the flux sum are in equilibrium. On the other hand, the flux and therefore the layer thickness may need adjustment.

Accounting for the second term of (5) in the calculation of the final thickness $\Delta p_k'^{fin}$ gives the simple equation :

$$\Delta p_k'^{fin} = \frac{\Delta p_k'^{\ddagger}}{p_b'^{\ddagger}} p_b'. \tag{27}$$

From this we obtain :

$$p_{k+1}'^{fin} = \sum_{l=1}^{k} \Delta p_l'^{fin} \quad \text{for} \quad k = 1, \ldots, N \tag{28}$$

with therefore : $p_{N+1}'^{fin} = p_b'^{fin} \equiv p_b'$. Then in the summations (6), (19), (23) and (28), we assume that the surface pressure is zero ($p_1' = 0$).

2.1.2 Interface diffusion

In the shallow-water multilayer model, the conservation of mass is cast in the form (BARAILLE & FILATOFF, 1995) :

$$\frac{\partial}{\partial t}\Delta p_k + \nabla \cdot (\mathbf{u}\Delta p)_k = 0 \tag{29}$$

Once the decomposition (1) is introduced in the formula, we can see that formally the equation (5) comes from the approximation: $(1 + \eta) \approx 1$. We notice that nevertheless the use of this approximation does not disturb in any case the property: $\partial p_b'/\partial t = 0$ (BARAILLE & FILATOFF, 1995). Thus we see that the sum of the changes in $\Delta p'$ over the vertical in this approximation is such that at each point, p_b' stays constant. In practice, the accounting of η is such that :

$$1 + \eta = \frac{1}{\rho_r H} \sum_{k=1}^{N} \rho_k h_k \tag{30}$$

so as to restore the introduction of a diffusion term $\nabla \cdot (\nu \nabla \Delta p')$ in the conservation equation (5). As in finite differences, the product $\nu \partial(\Delta p)/\partial x$ is numerically equivalent to $u_d \delta p$ (where δp represents the growth of the thickness of a layer between two adjacent meshes), BLECK *et al.* (1992) introduce a "diffusion velocity" $u_d \equiv \nu/\Delta x$ (where Δx is the size of the mesh) to simulate the isopycnic diffusion. We typically have : $u_d = 0.5\ cm/s$ for the variable Δp.

From the preceding FCT scheme, we obtain $p_{i,k}'^{fin}$ and $p_{i-1,k}'^{fin}$, the pressures at the k^{th} density interface in 2 adjacent points with coordinates x_i and x_{i-1}. In these 2 points, the bottom pressures are respectively p_{b_i}' and $p_{b_{i-1}}'$. We make this statement concrete by the form :

$$D_{i-1/2,k} = \min\left\{ p_{b_i}' - p_{i,k}' \max\left[p_{i-1,k}' - p_{b_{i-1}}', \frac{u_d \Delta t}{\Delta x}\left(p_{i-1,k}' - p_{i,k}' \right) \right] \right\}. \tag{31}$$

During a time interval Δt, the variation of pressure at the interface k comes from the expression :

$$\frac{\partial p_{i,k}'}{\partial t} + \frac{\Delta x}{\Delta t} \ (\nabla \cdot \mathbf{D})_{i,k} = 0 \tag{32}$$

This last equation is then written for the interfaces $k = 2, \ldots, N$:

$$p_{i,k}'^{n+1} = p_{i,k}'^{fin} - \left(D_{i+1/2,k} - D_{i-1/2,k} \right) \tag{33}$$

in which we put :

$$\Delta p'^{n+1}_{i,k} = p'^{n+1}_{i,k+1} - p'^{n+1}_{i,k} \tag{34}$$

To remain coherent with the previous step, the flux retained at the two interfaces of the layer are written :

$$\begin{cases} \mathbf{F}^{fin}_{i+1/2,k-1} & = \mathbf{P}'^{fin}_{i+1/2,k-1} + \frac{1}{\Delta t}\mathbf{D}_{i+1/2,k} \\ \mathbf{F}^{fin}_{i+1/2,k} & = \mathbf{P}'^{fin}_{i+1/2,k} - \frac{1}{\Delta t}\mathbf{D}_{i+1/2,k} \end{cases}$$

2.2 Usage

In MICOM 2.6, the numerical calculation of the baroclinic continuity equation is performed by the subprogram :

`subroutine cnuity` (list of arguments)

2.2.1 Order of operations

The algorithm is based on a consecutive treatment of isopycnic layers. First, we initialize the variables necessary to calculate the flux integrals over the vertical coordinate :

```
c
      do 41 j=1,jj1
c
      do 74 l=1,isp(j)
      do 74 i=ifp(j,l),ilp(j,l)
   74 util3(i,j)=0.
c
      do 40 l=1,isu(j)
      do 40 i=ifu(j,l),ilu(j,l)
   40 utotn(i,j)=0.
c
      do 41 l=1,isv(j)
      do 41 i=ifv(j,l),ilv(j,l)
   41 vtotn(i,j)=0.
c
```

The first calculational step consists of computing the two components of upstream flux in applying (8) :

```
c
c --- compute low-order (diffusive) fluxes at old time level
c
      do 12 j=1,jj1
c
      do 11 l=1,isu(j)
      do 11 i=ifu(j,l),ilu(j,l)
      utotm(i,j)=(u(i,j,km)+ubavg(i,j,m))*scu(i)
```

```
      if (utotm(i,j).ge.0.)  then
      q=min(dp(i-1,j,kn),max(0.,depthu(i,j)-util3(i-1,j)))
      else
      q=min(dp(i ,j,kn),max(0.,depthu(i,j)-util3(i ,j)))
      end if
   11 uflux(i,j)=utotm(i,j)*q
c
      do 12 l=1,isv(j)
      do 12 i=ifv(j,l),ilv(j,l)
      vtotm(i,j)=(v(i,j,km)+vbavg(i,j,m))*scv(i)
      if (vtotm(i,j).ge.0.)  then
      q=min(dp(i,j-1,kn),max(0.,depthv(i,j)-util3(i,j-1)))
      else
      q=min(dp(i,j ,kn),max(0.,depthv(i,j)-util3(i,j )))
      end if
   12 vflux(i,j)=vtotm(i,j)*q
c
```

Then, we determine the diffusive solution :

```
c
c --- advance -dp- field using low-order (diffusive) flux values
c
      dpmin=999.
c
      do 19 j=1,jj1
      do 19 l=1,isp(j)
      do 19 i=ifp(j,l),ilp(j,l)
      dpold(i,j,k)=dp(i,j,kn)
      util3(i,j)=util3(i,j)+dp(i,j,kn)
      dp(i,j,kn)=dp(i,j,kn)-(uflux(i+1,j)-uflux(i,j)
     .  +vflux(i,j+1)-vflux(i,j))*delt1*scp2i(i)
   19 dpmin=min(dpmin,dp(i,j,kn))
c
      if (dpmin.lt.-onecm) then
      do 190 j=1,jj1
      do 190 l=1,isp(j)
      do 190 i=ifp(j,l),ilp(j,l)
      if (dp(i,j,kn).eq.dpmin)
     .  write (*,'(i9,'' i,j,k='',3i4,'' neg.  dp (m) in loop 19'',
     .  f9.2)') nstep,i,j,k,dpmin/onem
  190 continue
      end if
c
```

Following that, we add the second order flux and calculate the anti-diffusive flux by simply taking the difference between the upstream flux and the second order flux. The second order

fluxes are stored in the arrays `uflux(i,j)` and `vflux(i,j)`. Meanwhile, we keep track of upstream fluxes in the arrays `uflx(i,j)` and `vflx(i,j)`. Note that the centering in space of the pressure term of equation (9) has been performed in the preceding calculation of the barotropic component of the current. (subroutine `barotp`). The numerical values of the thicknesses at the calculational points of the 2 horizontal components of velocity are found in the global variables `dpu(i,j,k)` and `dpv(i,j,k)`.

```
c
c --- compute 'antidiffusive' (i.e., high-order minus low-order) fluxes.
c --- high-order fluxes are second-order in space, time-centered.
c
      do 21 j=1,jj1
c
      do 20 l=1,isu(j)
      do 20 i=ifu(j,l),ilu(j,l)
      uflx(i,j,k)=uflux(i,j)
   20 uflux(i,j)=utotm(i,j)*dpu(i,j,km)-uflux(i,j)
c
      do 21 l=1,isv(j)
      do 21 i=ifv(j,l),ilv(j,l)
      vflx(i,j,k)=vflux(i,j)
   21 vflux(i,j)=vtotm(i,j)*dpv(i,j,km)-vflux(i,j)
c
```

The third step decomposes into two parts :

1) Determination of the local extrema of the variable Δp. We save the results in the arrays `util1(i,j)` and `util2(i,j)` ;

2) Computation of flux correctors following the method outlined above. The results are also stored in the arrays `util1(i,j)` and `util2(i,j)`

```
c
c --- at each grid point, determine the ratio of the largest permissible
c --- pos.  (neg.)  change in -dp- to the sum of all incoming (outgoing) fluxes
c
      do 25 i=1,ii1
      do 25 l=1,jsp(i)
      do 25 j=jfp(i,l),jlp(i,l)
      ja=max(jfp(i,l),j-1)
      jb=min(jlp(i,l),j+1)
      util1(i,j)=max(dp(i,j,kn),dp(i,ja,kn),dp(i,jb,kn))
   25 util2(i,j)=min(dp(i,j,kn),dp(i,ja,kn),dp(i,jb,kn))
c
      do 27 j=1,jj1
      do 27 l=1,isp(j)
      do 27 i=ifp(j,l),ilp(j,l)
```

```
      ia=max(ifp(j,l),i-1)
      ib=min(ilp(j,l),i+1)
c
      util1(i,j)=max(util1(i,j),dp(ia,j,kn),dp(ib,j,kn))
      util2(i,j)=max(0.,
     .  min(util2(i,j),dp(ia,j,kn),dp(ib,j,kn)))
c
      util1(i,j)=(util1(i,j)-dp(i,j,kn))
     .  /((max(0.,uflux(i,j))-min(0.,uflux(i+1,j))
     .  +max(0.,vflux(i,j))-min(0.,vflux(i,j+1))+epsil)
     .  *delt1*scp2i(i))
c
   27 util2(i,j)=(util2(i,j)-dp(i,j,kn))
     .  /((min(0.,uflux(i,j))-max(0.,uflux(i+1,j))
     .  +min(0.,vflux(i,j))-max(0.,vflux(i,j+1))-epsil)
     .  *delt1*scp2i(i))
c
```

In a fourth phase, we determine the flux correction given by (17). We store the results in variables `utotn(i,j)` and `vtotn(i,j)`.

```
c
c --- limit antidiffusive fluxes
c --- (keep track in -utotn,vtotn- of discrepancy between high-order
c --- fluxes and the sum of low-order and clipped antidiffusive fluxes.
c --- this will be used later to restore nondivergence of barotropic flow)
c
      do 29 j=1,jj1
c
      do 28 l=1,isu(j)
      do 28 i=ifu(j,l),ilu(j,l)
      if (uflux(i,j).ge.0.)  then
      clip=min(1.,util1(i,j),util2(i-1,j))
      else
      clip=min(1.,util2(i,j),util1(i-1,j))
      end if
      utotn(i,j)=utotn(i,j)+uflux(i,j)*(1.-clip)
      uflux(i,j)=uflux(i,j)*clip
   28 uflx(i,j,k)=uflx(i,j,k)+uflux(i,j)
c
      do 29 l=1,isv(j)
      do 29 i=ifv(j,l),ilv(j,l)
      if (vflux(i,j).ge.0.)  then
      clip=min(1.,util1(i,j),util2(i,j-1))
      else
      clip=min(1.,util2(i,j),util1(i,j-1))
      end if
```

```
      vtotn(i,j)=vtotn(i,j)+vflux(i,j)*(1.-clip)
      vflux(i,j)=vflux(i,j)*clip
   29 vflx(i,j,k)=vflx(i,j,k)+vflux(i,j)
c
```

Then we proceed to the inference of a new thickness given by (18).

```
c
c --- evaluate effect of antidiffusive fluxes on -dp- field
c
      dpmin=999.
c
      do 1 j=1,jj1
      do 1 l=1,isp(j)
      do 1 i=ifp(j,l),ilp(j,l)
      dp(i,j,kn)=dp(i,j,kn)-(uflux(i+1,j)-uflux(i,j)
     .  +vflux(i,j+1)-vflux(i,j))*delt1*scp2i(i)
      p(i,j,k+1)=p(i,j,k)+dp(i,j,kn)
    1 dpmin=min(dpmin,dp(i,j,kn))
c
      if (dpmin.lt.-onecm) then
      do 3 j=1,jj1
      do 3 l=1,isp(j)
      do 3 i=ifp(j,l),ilp(j,l)
      if (dp(i,j,kn).eq.dpmin)
     .  write (*,'(i9,'' i,j,k='',3i4,'' neg.  dp (m) in loop 1'',
     .  f9.2)') nstep,i,j,k,dpmin/onem
    3 continue
      end if
c
```

From there, it is then possible to determine the second correction to the flux expressed by (20), and to perform the estimation (22) of the thickness of the layer considered. The final values of the fluxes are placed in the arrays `uflx(i,j)` and `vflx(i,j)`.

```
c
c --- restore nondivergence of vertically integrated mass flow by
c --- recovering fluxes lost in the flux limiting process.
c --- treat these fluxes as an 'upstream' barotropic correction to
c --- the sum of diffusive and antidiffusive fluxes obtained so far.
c
      do 77 k=1,kk
      km=k+mm
      kn=k+nn
c
      do 45 j=1,jj1
c       do 44 l=1,isu(j)
```

```
      do 44 i=ifu(j,l),ilu(j,l)
      if (utotn(i,j).ge.0.)  then
      q=dp(i-1,j,kn)/p(i-1,j,kk+1)
      else
      q=dp(i ,j,kn)/p(i ,j,kk+1)
      end if
      uflux(i,j)=utotn(i,j)*q
   44 uflx(i,j,k)=uflx(i,j,k)+uflux(i,j)
c
      do 45 l=1,isv(j)
      do 45 i=ifv(j,l),ilv(j,l)
      if (vtotn(i,j).ge.0.)  then
      q=dp(i,j-1,kn)/p(i,j-1,kk+1)
      else
      q=dp(i,j ,kn)/p(i,j ,kk+1)
      end if
      vflux(i,j)=vtotn(i,j)*q
   45 vflx(i,j,k)=vflx(i,j,k)+vflux(i,j)
c
      dpmin=999.
c
      do 14 j=1,jj1
      do 14 l=1,isp(j)
      do 14 i=ifp(j,l),ilp(j,l)
      dp(i,j,kn)=dp(i,j,kn)-(uflux(i+1,j)-uflux(i,j)
     .  +vflux(i,j+1)-vflux(i,j))*delt1*scp2i(i)
      p(i,j,k+1)=p(i,j,k)+dp(i,j,kn)
   14 dpmin=min(dpmin,dp(i,j,kn))
c
      if (dpmin.lt.-onecm) then
      do 140 j=1,jj1
      do 140 l=1,isp(j)
      do 140 i=ifp(j,l),ilp(j,l)
      if (dp(i,j,kn).gt.dpmin) go to 140
      write (*,'(i9,'' i,j,k='',3i4,'' neg.  dp (m) in loop 14'',
     .  f9.2)') nstep,i,j,k,dpmin/onem
c
c --- if -dpmin exceeds 1 meter, provide additional diagnostics
      if (dpmin.gt.-onem) go to 140
      call prtij(i,j,k,u(1,1,km),1.,v(1,1,km),1.,dp(1,1,km),
     .  1./onem,temp(1,1,km),10.,saln(1,1,km),10.)
      call prtij(i,j,0,ubavg,1.,vbavg,1.,p(1,1,kk+1),1./onem,
     .  pbavg,1./onecm,p,1./onem)
  140 continue
      end if
c
   77 continue
```

```
c
```

For the last step, we make the adjustment (27).

```
c
c --- add bottom-pressure restoring term arising from split-explicit treatment
c --- of continuity equation (step 4 in appendix b to 1992 brhs paper)
c
      do 39 j=1,jj1
      do 39 l=1,isp(j)
      do 39 k=1,kk
      kn=k+nn
      do 39 i=ifp(j,l),ilp(j,l)
      dp(i,j,kn)=dp(i,j,kn)*pbot(i,j)/p(i,j,kk+1)
   39 p(i,j,k+1)=p(i,j,k)+dp(i,j,kn)
c
```

The treatment of the diffusion term consists of two principal steps :

1) To start, we proceed to calculate the interfacial diffusive flux given by the form (31), and then to correct the new flux with the aid of the expressions given in (35). The results are stored in the arrays `uflx(i,j)` and `vflx(i,j)`.

```
c
c --- ------------------------------------------------------------
c --- thickness diffusion (literally, interface depth diffusion)
c --- ------------------------------------------------------------
c
      do 816 k=2,kk
      kn=k+nn
c
      do 818 j=1,jj1
c
      do 817 l=1,isu(j)
      do 817 i=ifu(j,l),ilu(j,l)
      flxhi= .25*(p(i ,j,kk+1)-p(i ,j,k))*scp2(i )
      flxlo=-.25*(p(i-1,j,kk+1)-p(i-1,j,k))*scp2(i-1)
      uflux(i,j)=min(flxhi,max(flxlo,
     .  thkdff*delt1*(p(i-1,j,k)-p(i,j,k))*scu(i)))
      uflx(i,j,k-1)=uflx(i,j,k-1)+uflux(i,j)/delt1
      uflx(i,j,k )=uflx(i,j,k )-uflux(i,j)/delt1
  817 continue
c
      do 818 l=1,isv(j)
      do 818 i=ifv(j,l),ilv(j,l)
```

```
      flxhi= .25*(p(i,j ,kk+1)-p(i,j ,k))*scp2(i )
      flxlo=-.25*(p(i,j-1,kk+1)-p(i,j-1,k))*scp2(i )
      vflux(i,j)=min(flxhi,max(flxlo,
     .  thkdff*delt1*(p(i,j-1,k)-p(i,j,k))*scv(i)))
      vflx(i,j,k-1)=vflx(i,j,k-1)+vflux(i,j)/delt1
      vflx(i,j,k )=vflx(i,j,k )-vflux(i,j)/delt1
  818 continue
c
      do 816 j=1,jj1
      do 816 l=1,isp(j)
      do 816 i=ifp(j,l),ilp(j,l)
c
      p(i,j,k)=p(i,j,k)-(uflux(i+1,j)-uflux(i,j)
     .  +vflux(i,j+1)-vflux(i,j))*scp2i(i)
  816 dp(i,j,kn-1)=p(i,j,k)-p(i,j,k-1)
c
      do 819 j=1,jj1
      do 819 l=1,isp(j)
      do 819 i=ifp(j,l),ilp(j,l)
  819 dp(i,j,kk+nn)=p(i,j,kk+1)-p(i,j,kk)
c
```

2.2.2 Flowchart

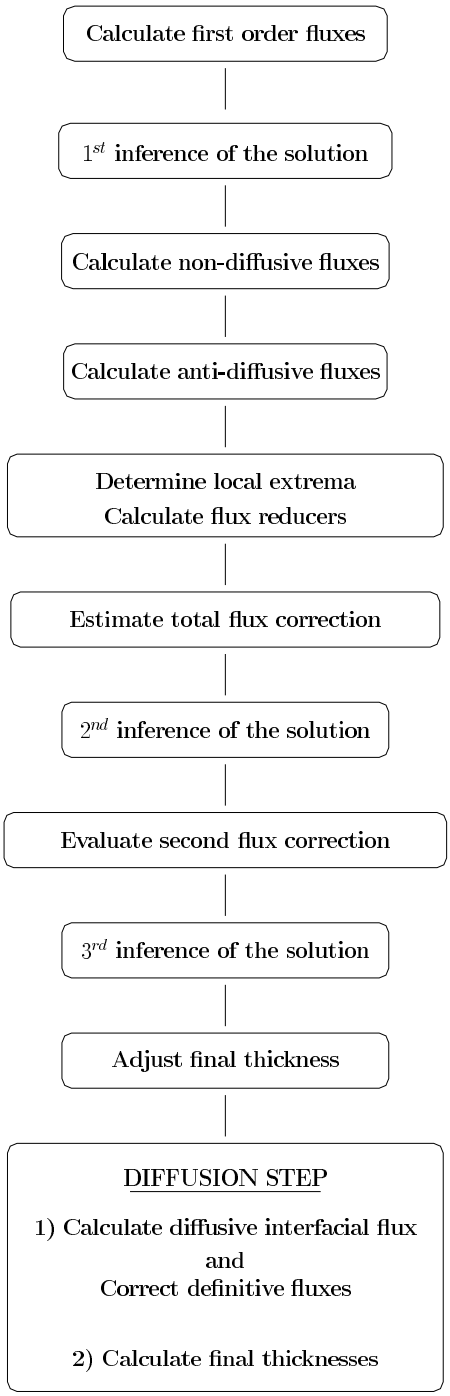

Figure 3: *Order of the treatment of the continuity equation for the baroclinic mode in MICOM 2.6*

2.3 Variables

2.3.1 Identification

Notation in the theory	Notation in **cnuity.f**
$C_{i+1/2}$	`clip`
$p'_{b_i} - p'_{i,k}$	`flxhi`
$p'_{i-1,k} - p'_{b_{i-1}}$	`flxlo`
u, v	`utotm(i,j), vtotm(i,j)`
$\overline{u}, \overline{v}$	`ubavg(i,j,n), vbavg(i,j,n)`
u', v'	`u(i,j,n), v(i,j,n)`
$\left(\Delta p_i'^{mid} + \Delta p_{i+1}'^{mid}\right)/2$	`dpu(i,j,k), dpv(i,j,k)`

2.3.2 Global variables

integer variables : `ifp(j,m)` `ilp(j,m)`, `ifu(j,m)`, `ifv(j,m)`, `ii1`, `ilu(j,m)`, `ilv(j,m)`, `isp(j)`, `isu(j)`, `isv(j)`, `jj1`, `kk`

real variables : `delt1`, `depthu(i,j)`, `depthv(i,j)`, `dp(i,j,k)`, `dpold(i,j,k)`, `dpu(i,j,k)`, `dpv(i,j,k)`, `epsil`, `onecm`, `onem`, `p(i,j,k)`, `saln(i,j,k)`, `scu(i)`, `scv(i)`, `scp2(i)`, `scp2i(i)`, `temp(i,j,k)`, `thkdff`, `ubavg(i,j,n)`, `uflux(i,j)`, `uflx(i,j,k)`, `util1(i,j)`, `util2(i,j)`, `util3(i,j)`, `utotm(i,j)`, `utotn(i,j)`, `vbavg(i,j,n)`, `vflux(i,j)`, `vflx(i,j,k)`, `vtotm(i,j)`, `vtotn(i,j)`

2.3.3 Local variables

`clip`	correction factor for antidiffusive fluxes
`dpmin`	variations in p'_b due to different corrections to the upstream flux
`flxhi,flxlo`	differences in pressure between the interfaces of a lyer and the bottom
`ia,ib,ja,jb`	intermediate indices
`q`	intermediate variable used in the calculation of the fluxes

2.4 Procedures

Subroutines `prtij`

3 Advection-diffusion : tsadvc.f

3.1 Formalism and numerical methods

In isopycnic coordinates, the thermal evolution equation takes the form :

$$\frac{\partial}{\partial t}T\Delta p + \underbrace{\nabla\cdot(\mathbf{u}T\Delta p)}_{advec} + \underbrace{\left(\dot{s}\frac{\partial p}{\partial s}T\right)_{bot} - \left(\dot{s}\frac{\partial p}{\partial s}T\right)_{top}}_{dia-diff} = \underbrace{\nabla\cdot(\nu\Delta p\,\nabla T)}_{iso-diff} + \mathcal{H}_T. \tag{35}$$

Δp is the thickness of layer k of temperature T. The radiative exchanges are represented by the term $\mathcal{H}_T$. The expression $(\dot{s}\partial p/\partial s)$ represents a vertical mass flux.
In MICOM, the problem of advection of heat and salt is treated by MPDATA coming from the work of Smolarkiewicz & Clark (1986) and Smolarkiewicz & Grabowski (1990). The diapycnic diffusion is accounted for in the diapycnic mixing algorithm described in Section 8. Regarding the isopycnic diffusion, the numerical method we are using is based on the different procedures we commented on in Section 3.1.3.

3.1.1 Maintaining the positivity of thickness

Consider the simple form of the problem of advection of an isopycnic layer of thickness Δp, of temperature T and driven by a horizontal velocity $\mathbf{u}$:

$$\frac{\partial}{\partial t}T\Delta p + \nabla\cdot(\mathbf{u}T\Delta p) = 0 \tag{36}$$

From the solution of the continuity equation, for each layer, we get the mass flows $(u\Delta p)^{n+1}_{i,j}$ and $(v\Delta p)^{n+1}_{i,j}$ as well as a diagnostic value for $\Delta p^{n+1}_{i,j}$. Let $\mathbf{P} = \mathbf{u}\Delta p$ and let h be the thickness of the layer of interest. Considering the conservation equation :

$$\frac{\partial h}{\partial t} + \nabla\cdot\mathbf{P} = 0 \tag{37}$$

and determining the variation δh follows from this form, we have :

$$\delta h(i,j) = -\frac{\Delta t}{\Delta x}\left[Px^{n+1}_{i+1,j} - Px^{n+1}_{i,j} + Py^{n+1}_{i,j+1} - Py^{n+1}_{i,j}\right] \tag{38}$$

Suppose the layer shrinks with time, *i.e.*, $\delta h < 0$. In this case, $|\delta h| > \Delta p^{n-1}$ brings about the unsatisfactory condition $\Delta p'^{n+1} < 0$. The fluxes resulting from the numerical treatment of the continuity equation are then inconsistent with the variations of thickness calculated in this same step. The mass flux as well as the layer thickness are necessary in the advection calculation. The mass fluxes are made to be consistent with the variation of layer thickness so that always in the case $\delta h < 0$, we have $|\delta h| \leq \Delta p^{n-1}$. Introduce the two utility thicknesses h_1 and h_2 :

$$h_1 = 1/2(\Delta p^{n+1} + \Delta p^{n-1} + \delta h) \;\; ; \;\; h_2 = 1/2(\Delta p^{n+1} + \Delta p^{n-1} - \delta h) \tag{39}$$

In the case where the fluxes are strictly consistent (*i.e.* : $\delta h = \Delta p^{n+1} - \Delta p^{n-1}$), we have :

$$h_1 = \Delta p^{n-1} \;\text{ and }\; h_2 = \Delta p^{n+1}$$

When the flux is consistent, we introduce the utility thicknesses given by (39). Otherwise, when the fluxes are inconsistent, we maintain the positivity of the layer thicknesses by setting :

$$h_1 = -\delta h \;\text{ and }\; h_2 = 0.$$

For $\delta h > 0$, we obtain the respective values of h_1 and h_2 with identical reasoning.

3.1.2 Treatment of the tendency term

To assure a gradual transition of Δp towards a null value, The finite difference expression of the tendency term in equation (35) that we generally write in the form :

$$\frac{T^{new}\Delta p^{new} - T^{old}\Delta p^{old}}{\Delta t}$$

is replaced by :

$$\frac{T^{new}(\Delta p^{new} + \epsilon) - T^{old}(\Delta p^{old} + \epsilon)}{\Delta t}. \tag{40}$$

The small parameter ϵ will be non-zero when the loss of mass during a time step averages at least 90% of the previous value. More precisely, we write :

$$\epsilon = A + (\epsilon_1^2 + A^2)^{1/2} \tag{41}$$

where $A = (0.1\Delta p^{old} - \Delta p^{new})/2$. The intermediate parameter ϵ_1 we therefore fix to the numerical value of 10 cm, introduced to assure the validity of (40) when Δp^{old} and Δp^{new} are tending toward zero.

3.1.3 Treatment of the diffusion term

As in finite differences, the product $\nu \partial T/\partial x$ is numerically equivalent to $u_d \delta T$ (where δT represents the temperature difference between the two adjacent mesh points bracketing u_d). BLECK *et al.* (1992) introduce a diffusion velocity $u_d \equiv \nu/\Delta x$ (where Δx is the size of the mesh) to simulate the isopycnic mixing. We typically have : $u_d = 1\ cm/s$ for the variables T and S. On the other hand, in the flux expression appearing in the diffusion term, the variation Δp is replaced by the harmonic average :

$$\widetilde{\Delta p} = \frac{2}{\Delta p_{i-1}^{-1} + \Delta p_i^{-1}} \tag{42}$$

The reasoning for this choice is not obvious. It can be found in BLECK *et al.* (1992). One can also understand it in the following way: Consider two neighboring mesh points to which are associated the two values Δp_{i-1} and Δp_i and the two temperatures T_{i-1} and T_i. In order for the introduction of the factor Δp in the flow expression not to have any effect, we will try to substitute a neutral value $\widetilde{\Delta p}$ that characterizes a neutral state. In this neutral context, two neighboring mesh points have the same amount of heat Q and an average temperature $\widetilde{T}$. If we allow that the turbulence and therefore the turbulent diffusion be interpreted as a mixing process, we simply obtain : $\widetilde{T} = (T_{i-1} + T_i)/2$. Of course the neutral state does not have the same thermal content Q. We therefore write :

$$\widetilde{T} = \frac{Q}{\widetilde{\Delta p}} = \frac{1}{2}\left(\frac{Q}{\Delta p_{i-1}} + \frac{Q}{\Delta p_i}\right) \tag{43}$$

an identity which led to the form (42). More over, as the inference of the new value $T_{i,j}^n$ requires a division by Δp, to treat the situations $\Delta p \to 0$, we introduce a residual thickness given the fixed numerical value of 1 mm.

3.1.4 Filtering

In the classical manner, to compensate for dispersion problems caused by the leapfrog scheme, we use Asselin filtering :

$$\widehat{T}^n\widehat{\Delta p}^n = \left[T^n(1-2\gamma)\Delta p^n + \gamma\left(\widehat{T}^{n-1}\widehat{\Delta p}^{n-1} + T^{n+1}\Delta p^{n+1}\right)\right] \tag{44}$$

So that this form remains valid when $\Delta p \to 0$, we introduce a residual thickness ϵ such that :

$$\widehat{T}^n = \left(\widehat{\Delta p}^n + \epsilon\right)^{-1}\left[T^n\left\{(1-2\gamma)\Delta p^n + \epsilon\right\} + \gamma\left(\widehat{T}^{n-1}\widehat{\Delta p}^{n-1} + T^{n+1}\Delta p^{n+1}\right)\right] \tag{45}$$

In application, ϵ takes a numerical value equivalent to 10^{-3} m. Concerning the thermodynamic variables, the value $\gamma = 0.015625$ is used. Moreover, in the code, the filtering is carried out in 2 steps. All things considered, we write :

$$\left[\widehat{T\delta p}\right]^n = T^n\left\{(1-2\gamma)\Delta p^n + \epsilon\right\} + \gamma\widehat{T}^{n-1}\widehat{\Delta p}^{n-1} \tag{46}$$

Then, after having calculated T^{n+1} and filtering the layer thickness Δp by the formula :

$$\widehat{\Delta p}^n = (1-2\gamma)\Delta p^n + \gamma\left(\widehat{\Delta p}^{n-1} + \Delta p^{n+1}\right) \tag{47}$$

we write in the second step :

$$\widehat{T}^n = \left(\widehat{\Delta p}^n + \epsilon\right)^{-1}\left\{\left[\widehat{T\delta p}\right]^n + \gamma T^{n+1}\Delta p^{n+1}\right\} \tag{48}$$

3.2 Usage

In the code of MICOM 2.6, the numerical calculation of the horizontal equations of advection-diffusion of heat and salt is realized by the subroutine :

`subroutine tsadvc` (argument list)

Concerning the mixed layer, the variables treated are the specific volume indicated by the variable `thmix(i,j,n)` and the salinity `saln(i,j,1)`. For the isopycnic layers, only the salinity is advected then diffused.

3.2.1 Order of operations

Concerning the mixed layer, we proceed to a smoothing of the components Px and Py of mass flux over three points. The results are put in the arrays `uflux(i,j)` and `vflux(i,j)`.

```
c
      if (k.eq.1) then
c --- ------------------------------------------------
c --- advection of thermodynamic variables in mixed layer
c --- ------------------------------------------------
c
c --- smooth mixed-layer mass fluxes in lateral direction
      do 791 j=1,jj1
      do 791 l=1,isv(j)
      do 791 i=ifv(j,l),ilv(j,l)
      ia=max(i-1,ifv(j,l))
      ib=min(i+1,ilv(j,l))
  791 vflux(i,j)=.5*vflx(i,j,1)+.25*(vflx(ia,j,1)+vflx(ib,j,1))
c
      do 792 i=1,ii1
      do 792 l=1,jsu(i)
      do 792 j=jfu(i,l),jlu(i,l)
      ja=max(j-1,jfu(i,l))
      jb=min(j+1,jlu(i,l))
  792 uflux(i,j)=.5*uflx(i,j,1)+.25*(uflx(i,ja,1)+uflx(i,jb,1))
c
```

Following this, for each computational point, we do the first phase of the filtering following the formula (46) and we start the procedure verifying the coherence of the mass fluxes with the variation of thickness of the surface layer. The variables h_1 and h_2 correspond to the arrays `util1(i,j)` and `util2(i,j)`.

```
c
      posdef=thbase
c
      do 491 j=1,jj1
      do 491 l=1,isp(j)
      do 491 i=ifp(j,l),ilp(j,l)
```

```
c
c --- time smoothing of mixed layer thermodynamic variables (part 1)
      pold=max(0.,dpold(i,j,k))
      pmid=max(0.,dp(i,j,km))
      saln(i,j,km)=saln(i,j,km)*(wts1*pmid+onemm)+wts2*saln(i,j,kn)*pold
      thmix(i,j,m)=thmix(i,j,m)*(wts1*pmid+onemm)+wts2*thmix(i,j,n)*pold
c
c --- before calling 'advem', make sure (a) mass fluxes are consistent
c --- with layer thickness change, and (b) all fields are positive-definite
      flxdiv=(uflux(i+1,j)-uflux(i,j)
     .  +vflux(i,j+1)-vflux(i,j))*delt1*scp2i(i)
      util2(i,j)=.5*(dpold(i,j,k)+dp(i,j,kn)-flxdiv)
      util1(i,j)=.5*(dpold(i,j,k)+dp(i,j,kn)+flxdiv)
      offset=min(0.,util1(i,j),util2(i,j))
      util2(i,j)=util2(i,j)-offset
      util1(i,j)=util1(i,j)-offset
c
      smin=min(smin,saln(i,j,kn))
      smax=max(smax,saln(i,j,kn))
      tmin=min(tmin,thmix(i,j,n))
      tmax=max(tmax,thmix(i,j,n))
  491 thmix(i,j,n)=thmix(i,j,n)+posdef
c

c
```

Then, the advection of the mixed layer characteristics is realized by the instructions :

```
c
      call advem(2,thmix(1,1,n),uflux,vflux,scp2,scp2i,
     .  delt1,util1,util2)
      call advem(2,saln(1,1,kn),uflux,vflux,scp2,scp2i,
     .  delt1,util1,util2)
c
```

After having filtered the layer thicknesses by the formula (101), we execute the double step of filtering given by the relation (48). We are then ready to diagnose negative characteristics.

```
c
      do 461 j=1,jj1
      do 461 l=1,isp(j)
      do 461 i=ifp(j,l),ilp(j,l)
      thmix(i,j,n)=thmix(i,j,n)-posdef
      sminn=min(sminn,saln(i,j,kn))
      smaxx=max(smaxx,saln(i,j,kn))
      tminn=min(tminn,thmix(i,j,n))
```

```
      tmaxx=max(tmaxx,thmix(i,j,n))
c
c --- time smoothing of thickness field
      pold=max(0.,dpold(i,j,k))
      pmid=max(0.,dp(i,j,km))
      pnew=max(0.,dp(i,j,kn))
      dp(i,j,km)=pmid*wts1+(pold+pnew)*wts2
c --- time smoothing of mixed layer thermodynamic variables (part 2)
      pmid=max(0.,dp(i,j,km))
      saln(i,j,km)=(saln(i,j,km)+wts2*saln(i,j,kn)*pnew)/(pmid+onemm)
  461 thmix(i,j,m)=(thmix(i,j,m)+wts2*thmix(i,j,n)*pnew)/(pmid+onemm)
c
      if (tminn+posdef.lt.0..or.sminn.lt.0.)
     .  write (*,'(i9,i4,'' neg.  thmix/saln after advem call '',
     .  3pf9.2,0pf9.2)') nstep,k,tminn,sminn
c
```

It remains to calculate the neutral thickness for the mesh by accounting for the effect of diffusion over the mixed layer characteristics :

```
c
c --- ------------------------------------------------
c --- diffusion of thermodynamic variables in mixed layer
c --- ------------------------------------------------
c
      do 451 j=1,jj1
c
      do 441 l=1,isu(j)
      do 441 i=ifu(j,l),ilu(j,l)
      factor=scu(i)*harmon(max(dp(i-1,j,kn),onemm)
     .  ,max(dp(i ,j,kn),onemm))
      uflux2(i,j)=factor*(saln(i-1,j,kn)-saln(i,j,kn))
  441 uflux3(i,j)=factor*(thmix(i-1,j,n)-thmix(i,j,n))
c
      do 451 l=1,isv(j)
      do 451 i=ifv(j,l),ilv(j,l)
      factor=scv(i)*harmon(max(dp(i,j-1,kn),onemm)
     .  ,max(dp(i,j ,kn),onemm))
      vflux2(i,j)=factor*(saln(i,j-1,kn)-saln(i,j,kn))
  451 vflux3(i,j)=factor*(thmix(i,j-1,n)-thmix(i,j,n))
c
      do 462 j=1,jj1
      do 462 l=1,isp(j)
      do 462 i=ifp(j,l),ilp(j,l)
      factor=temdff*delt1/(scp2(i)*max(dp(i,j,kn),onemm))
      saln(i,j,kn)=saln(i,j,kn)-(uflux2(i+1,j)-uflux2(i,j)
     .  +vflux2(i,j+1)-vflux2(i,j))*factor
```

```
      thmix(i,j,n)=thmix(i,j,n)-(uflux3(i+1,j)-uflux3(i,j)
     .  +vflux3(i,j+1)-vflux3(i,j))*factor
  462 temp(i,j,kn)=tofsig(thmix(i,j,n)+thbase,saln(i,j,kn))
c
```

Concerning the isopycnic layers, the order of operations is identical to that described for the mixed layer except for the fact that only the salinity is advected, then diffused. MICOM 2.6 is designed to treat in parallel any tracer problem :

```
c
c --- advection of 'ventilation' tracer (every 5 time steps only)
c
      if (trcout.and.mod(nstep,5).eq.0) then
      do 68 j=1,jj1
      do 68 l=1,isp(j)
      do 68 i=ifp(j,l),ilp(j,l)
      flxdiv=(uflx(i+1,j,k)-uflx(i,j,k)
     .  +vflx(i,j+1,k)-vflx(i,j,k))*5.*baclin*scp2i(i)
      util2(i,j)=.5*(dpold(i,j,k)+dp(i,j,kn)-flxdiv)
      util1(i,j)=.5*(dpold(i,j,k)+dp(i,j,kn)+flxdiv)
      offset=min(0.,util1(i,j),util2(i,j))
      util2(i,j)=util2(i,j)-offset
      util1(i,j)=util1(i,j)-offset
   68 tracer(i,j,k)=max(0.,tracer(i,j,k))
      call advem(2,tracer(1,1,k),uflx(1,1,k),vflx(1,1,k),scp2,scp2i,
     .  5.*baclin,util1,util2)
      end if
c
```

3.2.2 Flowchart

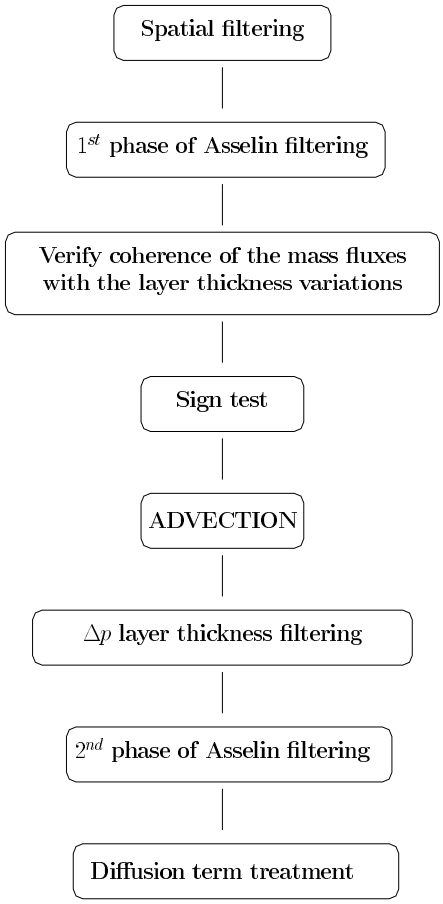

Figure 4: *Order of transport and horizontal diffusion calculations in MICOM 2.6*

3.3 Variables

3.3.1 Identification

Notation in the theory	Notation in **diapfl.f**
$\widetilde{\Delta p}$	`factor`
δh	`flxdiv`
h_1, h_2	`util1(i,j),util2(i,j)`
$\gamma, (1-2\gamma)$	`wts2,wts1`

3.3.2 Global variables

integer variables :	`ifp(j,m)` `ilp(j,m)`, `ifv(j,m)`, `ilv(j,m)`, `isp(j)`, `isu(j)`, `isv(j)`, `jflx(i,j)`, `jfu(i,m)`, `jlu(i,m)`, `jsu(i)`, `klist(i,j)`, `kk`
real variables :	`baclin`, `delt1`, `dp(i,j,k)`, `dpold(i,j,k)`, `saln(i,j,k)`, `scu(i)`, `scv(i)`, `scp2(i)`, `scp2i(i)`, `temdff`, `temp(i,j,k)`, `thmix(i,j,n)`, `tracer(i,j,k)`, `uflux(i,j)`, `uflux2(i,j)`, `uflux3(i,j)`, `uflx(i,j,k)`, `util1(i,j)`, `util2(i,j)`, `vflux(i,j)`, `vflux2(i,j)`, `vflux3(i,j)`, `vflx(i,j,k)`, `wts1`, `wts2`
logical variables :	`diagno`

3.3.3 local variables

`factor`	intermediate variable in the diffusion term calculation
`flxdiv`	variation of thickness δh due to the divergence of momentum flux
`ia,ib`	loop counters
`ja,jb`	loop counters
`offset`	intermediate variable in the calculation of h_1 and h_2
`posdef`	reference value of the specific volume in the mixed layer
`pmid,pnew,pold`	intermediate pressures
`smax,smaxx,smin,sminn`	intermediate salinities
`tmax,tmaxx,tmin,tminn`	intermediate temperatures

3.4 Procedures

Functions	`harmon`
Subroutines	`prtij`, `advem`

3.5 The SMOLARKIEWICZ MPDATA

3.5.1 Formalism

The formalism used by BLECK to treat the problem of advection of a non-negative quantity ψ in MICOM is very similar to that used by SMOLARKIEWICZ in MPDATA (SMOLARKIEWICZ & CLARK, 1986 and SMOLARKIEWICZ & GRABOWSKI, 1990). In a bidimensional form, the general equation in conservation form can be written :

$$\frac{\partial f\psi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{49}$$

where $(F, G) = (u\psi, v\psi)$ represent the fluxes of the quantity ψ in the two directions x, y. f is an arbitrary positive function. To preserve the sign of ψ, we approximate the form (49) with a

donor-cell scheme. In the first approximation, the upstream flux as well as the components of the velocity are defined at the interfaces between the cells. This flux can be generally expressed :

$$F^{up}_{i+1/2,j} = \frac{u_{i+1/2} + |u_{i+1/2}|}{2}\psi_{i,j} + \frac{u_{i+1/2} - |u_{i+1/2}|}{2}\psi_{i+1,j} \tag{50}$$

which, on introducing the two operators :

$$\overline{\psi}^x = (\psi_{i,j} + \psi_{i+1,j})/2 \quad \text{et} \quad \delta_x\psi = (\psi_{i+1,j} - \psi_{i,j})/\Delta x$$

then condenses to the form :

$$F^{up} = u\overline{\psi}^x - \frac{|u|\Delta x}{2}\delta_x\psi \tag{51}$$

Starting from the standard Taylor expansion :

$$\phi_{i\pm1/2,j} = \phi_{ij} \pm \frac{\Delta x}{2}\frac{\partial\phi}{\partial x} + \frac{\Delta x^2}{8}\frac{\partial^2\phi}{\partial x^2} + O(\Delta x^3)$$

where ϕ is a variable such that :

$$\overline{\phi}^x = \phi + O(\Delta x^2)$$

$$\delta_x\phi = \frac{\partial\phi}{\partial x} + O(\Delta x^2).$$

The introduction of the forms in (50) lead to :

$$\delta_x F^{up} = \frac{\partial}{\partial x}\left[u[\psi + O(\Delta x^2)] - \frac{|u|\Delta x}{2}[\frac{\partial\psi}{\partial x} + O(\Delta x^2)]\right] \tag{52}$$

Similarly, expanding in a Taylor series in the time step allows us to write the identity :

$$\frac{(f\psi)^{n+1} - (f\psi)^n}{\Delta t} \equiv \delta_t f\psi = \frac{\partial f\psi}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 f\psi}{\partial t^2} + O(\Delta t^2)$$

Taking into account (49), the second derivative of the last term can be expressed as a function of spatial derivatives :

$$\begin{aligned}
\frac{\partial^2 f\psi}{\partial t^2} &= -\frac{\partial}{\partial t}\left[\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right] \\
&= -\frac{\partial}{\partial x}\left(\frac{u}{f}\frac{\partial f\psi}{\partial t}\right) - \frac{\partial}{\partial y}\left(\frac{v}{f}\frac{\partial f\psi}{\partial t}\right) \\
&= \frac{\partial}{\partial x}\left[\frac{u}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{v}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right].
\end{aligned}$$

which gives :

$$\delta_t f\psi = \frac{\partial f\psi}{\partial t} + \frac{\Delta t}{2}\left\{\frac{\partial}{\partial x}\left[\frac{u}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{v}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right]\right\} + O(\Delta t^2) \tag{53}$$

In combining (52) and (53), we finally obtain :

$$\begin{aligned}
\delta_t f\psi + \delta_x F^{up} + \delta_y G^{up} &= \frac{\partial f\psi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} \\
&\quad + \frac{\Delta t}{2}\left\{\frac{\partial}{\partial x}\left[\frac{u}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{v}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right]\right\} \\
&\quad - \frac{\partial}{\partial x}\left[\frac{|u|\Delta x}{2}\frac{\partial\psi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\frac{|v|\Delta y}{2}\frac{\partial\psi}{\partial y}\right] + O(\Delta t^2, \Delta x^2, \Delta y^2)
\end{aligned}$$

The essence of the MPDATA scheme is to correct truncation errors of the forward-upstream scheme by introducing the divergence of the anti-diffusive fluxes :

$$F^{anti} = \frac{|u|\Delta x}{2}\frac{\partial \psi}{\partial x} - \frac{u\Delta t}{2f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)$$
$$G^{anti} = \frac{|v|\Delta y}{2}\frac{\partial \psi}{\partial y} - \frac{v\Delta t}{2f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right) \tag{54}$$

and to ameliorate the precision of the initial scheme in resolving (49) with the form :

$$\delta_t f\psi = -(\delta_x F^{up} + \delta_y G^{up}) - (\delta_x F^{anti} + \delta_y G^{anti}). \tag{55}$$

In the numerical calculations, the final solution is evaluated in two steps :

1) Determination of the upstream fluxes F^{up} and G^{up} and approximation of the transported but diffused solution ψ^{up};

2) Establishment of the anti-diffusive fluxes. At this phase, the positivity is assured by submitting the fluxes F^{anti} and G^{anti} to a procedure of flux reduction identical to the one implemented in the algorithm of Flux Corrected Transport (*cf.* § 2.1.1). Since we use a C grid to spatially discretize the different variables, this step requires spatially-centered finite differences when we approximate the anti-diffusive fluxes given by (54). So we can write :

$$F_c^{up}(i,j) = \frac{1}{2}u(i,j)\left[\psi^{up}(i-1,j) + \psi^{up}(i,j)\right]$$
$$G_c^{up}(i,j) = \frac{1}{2}v(i,j)\left[\psi^{up}(i,j-1) + \psi^{up}(i,j)\right] \tag{56}$$

then :

$$\frac{\partial F}{\partial x} \equiv \frac{1}{\Delta x}\left[F_c^{up}(i+1,j) + F_c^{up}(i,j)\right]$$
$$\frac{\partial G}{\partial y} \equiv \frac{1}{\Delta y}\left[G_c^{up}(i,j+1) + G_c^{up}(i,j)\right] \tag{57}$$

Putting $Div\mathcal{F} = \partial F/\partial x + \partial G/\partial y$ and setting $f = 1$, we then write :

$$F_c^{anti} = \frac{1}{2}|u(i,j)|\left[\psi^{up}(i,j) - \psi^{up}(i-1,j)\right] - \frac{\Delta t}{4}u(i,j)\left[Div\mathcal{F}(i-1,j) + Div\mathcal{F}(i,j)\right]$$
$$G_c^{anti} = \frac{1}{2}|v(i,j)|\left[\psi^{up}(i,j) - \psi^{up}(i,j-1)\right] - \frac{\Delta t}{4}v(i,j)\left[Div\mathcal{F}(i,j-1) + Div\mathcal{F}(i,j)\right] \tag{58}$$

3.5.2 Usage

The usage of the third order numerical scheme for advection of heat and salt, based on the work of SMOLARKIEWICZ & CLARK (1986) and SMOLARKIEWICZ & GRABOWSKI (1990), is implemented in the subroutine :

```
subroutine advem (iord,fld,u,v,scal,scali,dt,fco,fc)
```

The transported variable is `fld`. The arguments u and v represent the mass fluxes $u\Delta p$ and $v\Delta p$. Their numerical values come from solution of the continuity equation. The two arguments `fco` and `fc` correspond to the two utility thicknesses h_1 and h_2 of the layer considered, as we understood in §3.1.1. They are needed in the subroutine to be able to apply the form (40) and therefore to determine ψ^{up} from which we eventually calculate the finite-difference expressions for the anti-diffusive fluxes. For `iord=1`, the numerical scheme returns a simple donor-cell. When the transport variable `fld` is worked out using the quantities $u\Delta p$ and $v\Delta p$ and not using the characteristics u and v of the velocity field, the calculation of a divergence of the anti-diffusive fluxes centered in time implies also storing the numerical values of the fluxes obtained by (58) with the quantities of average thickness :

$$\overline{\overline{\Delta p}} = \left[h_1(i,j,n-1) + h_2(i,j,n-1) + h_1(i,j,n) + h_2(i,j,n)\right]/4 \tag{59}$$

3.5.3 Order of operations

The algorithm includes a preliminary step which consists of determining one part, the local extrema, and another part, the fluxes F^{up} and G^{up} according to the upstream method.

```
c
c --- compute low-order fluxes and maxima/minima to be used in flux clipping
c
      do 2 j=1,jj1
c
      do 22 l=1,isp(j)
      flx(ifp(j,l) ,j)=0.
      flx(ilp(j,l)+1,j)=0.
c
      do 22 i=ifp(j,l),ilp(j,l)
      ip1=min(i+1,ilp(j,l))
      im1=max(i-1,ifp(j,l))
      fmx(i,j)=max(fld(i,j),fld(im1,j),fld(ip1,j))
   22 fmn(i,j)=min(fld(i,j),fld(im1,j),fld(ip1,j))
c
      do 2 l=1,isu(j)
      do 2 i=ifu(j,l),ilu(j,l)
      if (u(i,j).ge.0.)  then
      q=fld(i-1,j)
      else
      q=fld(i ,j)
      end if
    2 flx(i,j)=u(i,j)*q
```

```
c
      do 3 i=1,ii1
c
      do 33 l=1,jsp(i)
      fly(i,jfp(i,l) )=0.
      fly(i,jlp(i,l)+1)=0.
c
      do 33 j=jfp(i,l),jlp(i,l)
      jp1=min(j+1,jlp(i,l))
      jm1=max(j-1,jfp(i,l))
      fmx(i,j)=max(fmx(i,j),fld(i,jm1),fld(i,jp1))
   33 fmn(i,j)=min(fmn(i,j),fld(i,jm1),fld(i,jp1))
c
      do 3 l=1,jsv(i)
      do 3 j=jfv(i,l),jlv(i,l)
      if (v(i,j).ge.0.)  then
      q=fld(i,j-1)
      else
      q=fld(i,j )
      end if
    3 fly(i,j)=v(i,j)*q
c
```

The following step returns an estimate of the divergence of the flux and enforces monotonicity. In this first phase of inferring a new value of the variable `fld` from the form (40), we introduce the parameter ϵ as defined in §3.1.2.

```
c
      do 61 j=1,jj1
      do 61 l=1,isp(j)
      do 61 i=ifp(j,l),ilp(j,l)
      flxdiv(i,j)=(flx(i+1,j)-flx(i,j)+fly(i,j+1)-fly(i,j))*dt*scali(i)
      cushn=.5*(.1*fco(i,j)-fc(i,j))
      cushn=cushn+sqrt(tencm*tencm+cushn*cushn)
      fld(i,j)=(fld(i,j)*(cushn+fco(i,j))-flxdiv(i,j))
     .  /(cushn+fc (i,j))
c --- don't let -fld- value go outside range of surrounding values
   61 fld(i,j)=max(fmn(i,j),min(fld(i,j),fmx(i,j)))
c
```

Next comes the calculation of the anti-diffusive fluxes with the evaluation of the fluxes F_c^{up} and G_c^{up} from the expressions given by (56). We generate an estimation of the fluxes F_c^{anti} and G_c^{anti} by (58). The results are stored in the arrays `flx(i,j)` and `fly(i,j)`.

```
c
c --- compute antidiffusive fluxes
c
```

```
      do 4 j=1,jj1
c
      do 44 l=1,isp(j)
      do 44 i=ifp(j,l),ilp(j,l)
      ip1=min(i+1,ilp(j,l))
      im1=max(i-1,ifp(j,l))
      fmx(i,j)=max(fmx(i,j),fld(im1,j),fld(ip1,j))
   44 fmn(i,j)=min(fmn(i,j),fld(im1,j),fld(ip1,j))
c
      do 4 l=1,isu(j)
      do 4 i=ifu(j,l),ilu(j,l)
    4 flx(i,j)=u(i,j)*.5*(fld(i-1,j)+fld(i,j))
c
      do 5 i=1,ii1
c
      do 55 l=1,jsp(i)
      do 55 j=jfp(i,l),jlp(i,l)
      jp1=min(j+1,jlp(i,l))
      jm1=max(j-1,jfp(i,l))
      fmx(i,j)=max(fmx(i,j),fld(i,jm1),fld(i,jp1))
   55 fmn(i,j)=min(fmn(i,j),fld(i,jm1),fld(i,jp1))
c
      do 5 l=1,jsv(i)
      do 5 j=jfv(i,l),jlv(i,l)
    5 fly(i,j)=v(i,j)*.5*(fld(i,j-1)+fld(i,j))
c
      do 66 j=1,jj1
      do 66 l=1,isp(j)
      do 66 i=ifp(j,l),ilp(j,l)
   66 flxdiv(i,j)=(flx(i+1,j)-flx(i,j)+fly(i,j+1)-fly(i,j))*dt*scali(i)
c
      do 8 j=1,jj1
c
      do 7 l=1,isu(j)
      do 7 i=ifu(j,l),ilu(j,l)
    7 flx(i,j)=.5*abs(u(i,j))*(fld(i,j)-fld(i-1,j))
     .  -u(i,j)*(flxdiv(i,j)+flxdiv(i-1,j))
     .  /(fco(i,j)+fco(i-1,j)+fc(i,j)+fc(i-1,j)+epsil)
c
      do 8 l=1,isv(j)
      do 8 i=ifv(j,l),ilv(j,l)
    8 fly(i,j)=.5*abs(v(i,j))*(fld(i,j)-fld(i,j-1))
     .  -v(i,j)*(flxdiv(i,j)+flxdiv(i,j-1))
     .  /(fco(i,j)+fco(i,j-1)+fc(i,j)+fc(i,j-1)+epsil)
c
```

The numerical values obtained are then subjected to the flux reduction method.

```
c
c---- limit antidiffusive fluxes
c
      do 16 j=1,jj1
      do 16 l=1,isp(j)
      do 16 i=ifp(j,l),ilp(j,l)
      flp(i,j)=(fmx(i,j)-fld(i,j))*fc(i,j)*scal(i)/dt / ( epsil
     .  -min(0.,flx(i+1,j))+max(0.,flx(i,j))
     .  -min(0.,fly(i,j+1))+max(0.,fly(i,j)) )
      fln(i,j)=(fld(i,j)-fmn(i,j))*fc(i,j)*scal(i)/dt / ( epsil
     .  +max(0.,flx(i+1,j))-min(0.,flx(i,j))
     .  +max(0.,fly(i,j+1))-min(0.,fly(i,j)) )
   16 continue
c
      do 18 j=1,jj1
c
      do 17 l=1,isu(j)
      do 17 i=ifu(j,l),ilu(j,l)
      flx(i,j)=max(0.,flx(i,j))*min(1.,flp(i,j),fln(i-1,j))
     .  +min(0.,flx(i,j))*min(1.,flp(i-1,j),fln(i,j))
   17 continue
c       do 18 l=1,isv(j)
      do 18 i=ifv(j,l),ilv(j,l)
      fly(i,j)=max(0.,fly(i,j))*min(1.,flp(i,j),fln(i,j-1))
     .  +min(0.,fly(i,j))*min(1.,flp(i,j-1),fln(i,j))
   18 continue
c
```

The last operation consists of calculating the divergence of these fluxes from which we obtain the final value of the transported variable, taking into account, again, the expression (40).

```
c
      do 62 j=1,jj1
      do 62 l=1,isp(j)
      do 62 i=ifp(j,l),ilp(j,l)
      flxdiv(i,j)=(flx(i+1,j)-flx(i,j)+fly(i,j+1)-fly(i,j))*dt*scali(i)
      cushn=.5*(.1*fco(i,j)-fc(i,j))
      cushn=cushn+sqrt(tencm*tencm+cushn*cushn)
      fld(i,j)=(fld(i,j)*(cushn+fc(i,j))-flxdiv(i,j))
     .  /(cushn+fc(i,j))
c --- don't let -fld- value go outside range of surrounding values
   62 fld(i,j)=max(fmn(i,j),min(fld(i,j),fmx(i,j)))
c
```

Flowchart

3.5.4 Flowchart

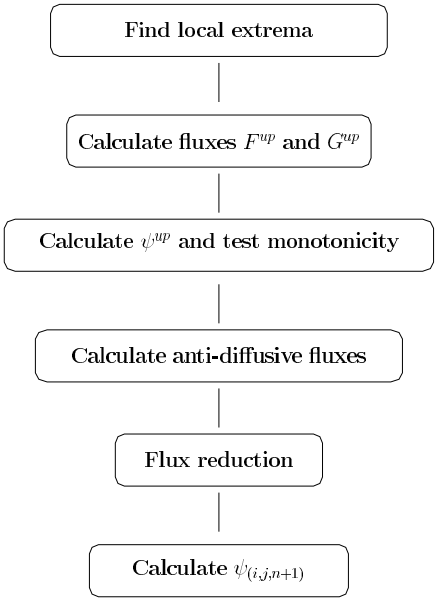

Figure 5: *Order of horizontal transport calculations in MICOM 2.6*

3.5.5 Variables

Identification

Notation in the theory	Notation in **advem.f**
T, S	`fld(i,j)`
ϵ	`cushn`
h_1, h_2	`fco(i,j),fc(i,j)`
F, G	`flx(i,j),fly(i,j)`
$Div\mathcal{F}$	`flxdiv(i,j)`
$u\Delta p, v\Delta p$	`u(i,j),v(i,j)`
$1/\Delta x$	`scali(i)`

Global variables

integer variables : `ifp(j,m)` `ilp(j,m)`, `ifu(j,m)` `ilu(j,m)`, `isp(j)`, `isu(j)`, `isv(j)`, `jfp(j,m)`, `jfv(j,m)`, `jlp(j,m)`, `jlv(j,m)`, `jsp(j)`, `jsv(j)`

real variables : `delt1`, `saln(i,j,k)`, `scp2(i)`, `scp2i(i)`, `thmix(i,j,n)`, `uflux(i,j)`, `util1(i,j)`, `util2(i,j)`, `vflux(i,j)`,

The different arrays that store the intermediate results are introduced by the statement :

```
common/work/fmx(idm,jdm),fmn(idm,jdm),flp(idm,jdm),fln(idm,jdm),flx(idm,jdm),
fly(idm,jdm),flxdiv(idm,jdm)
```

Local variables

`cushn`	mass loss parameter
`fco(i,j),fc(i,j)`	utility thicknesses
`fld(i,j)`	constituent
`fln(i,j),flp(i,j)`	flux reduction factors
`flx(i,j),fly(i,j)`	flux of constituent in each direction x and y
`fmn(i,j),fmx(i,j)`	extrema of `fld` in a nearby neighborhood
`iord`	order of approximation of the numerical scheme
`ip1,im1`	intermediate indices
`jp1,jm1`	intermediate indices
`q`	variable representing a constituent

`scal(i)`	Δx for each calculational row
`scali(i)`	inverse of Δx
`u(i,j), v(i,j)`	components of mass flux at node (i, j)

4 Forcing : momeq1.f

4.1 Formalism and numerical techniques

In an isopycnic system, the momentum conservation equation is of the form :

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \frac{\mathbf{u}^2}{2} + (\zeta + f)\mathbf{k} \times \mathbf{u} = -\nabla M - g\frac{\partial \boldsymbol{\tau}}{\partial p} \tag{60}$$

with :

ζ : relative vorticity
$\mathbf{k}$: vertical unit vector
M : Montgomery potential
$\boldsymbol{\tau}$: Reynolds stress
f : Coriolis parameter

The subroutine `momeq1` computes the following forcing terms of the momentum equation (60) :

1) the Montgomery potential;
2) the surface wind effects;
3) the bottom drag.

4.1.1 Montgomery potential

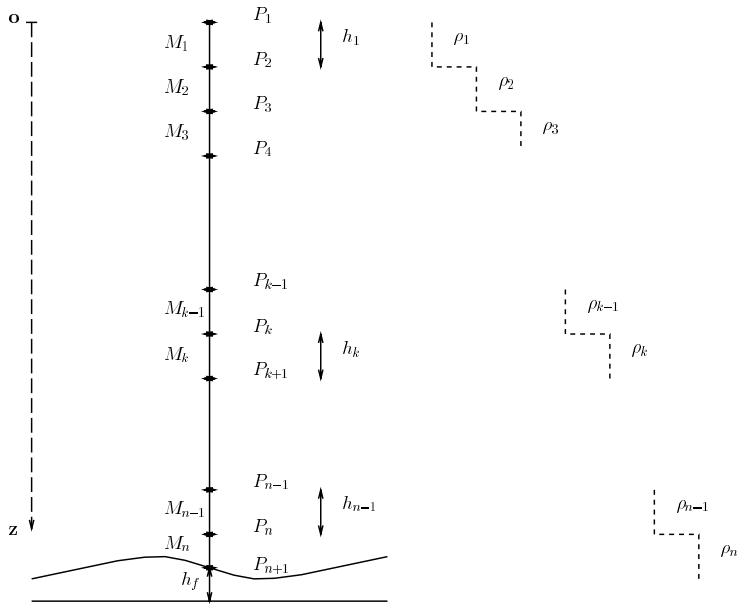

Figure 6: *Initial vertical distribution of pressure, density, and Montgomery potential*

To the initial distribution (figure 6), in MICOM 2.6, we add the surface condition $p_1^{init} = 0$. The portion of the ocean represented is at rest and we neglect the effects of the atmospheric pressure gradient. If, theoretically, the Montgomery potential is expressed in the general form $M = p + \rho gz$, it is not self-explanatory. However, a clear physical interpretation comes from letting Π_k be the potential that represents, along the vertical, the pressure deficit caused by the fact that the water column above the considered layer k has a density that is different from the one of this layer. We assume the hydrostatic hypothesis. Over the initial profile, for the layer k, we then easily express the pressure deficit by the recurrence relation :

$$\Pi_{k+1} = \Pi_k - g(\rho_{k+1} - \rho_k) \sum_{i=1}^{k} \langle h_k \rangle \tag{61}$$

As an initial condition, we put $\Pi_1 = 0$. In the bottom layer, where the density goes to ρ_N, we obtain by this relation Π_N.

When this stratified ocean is in motion, the height of the total column of water, not greater than H, is D such that :

$$D = \sum_{k=1}^{N} h_k \tag{62}$$

In the moving ocean, the Montgomery potential M is equivalent to the potential Π established by the initial state. So the Montgomery potential in layer k is given by the recurrence relation :

$$M_k = M_{k+1} + g(\rho_{k+1} - \rho_k) \sum_{i=1}^{k} h_k \tag{63}$$

As η is independent of the layer K, we also have :

$$M_k = M_{k+1} + g(1+\eta)(\rho_{k+1} - \rho_k) \sum_{i=1}^{k} h'_k \tag{64}$$

To connect the potential Π_k to the Montgomery potential M_k present in the baroclinic momentum equations (*cf.* § 5), we start at the bottom. We assume a homogeneous ocean of density ρ_N. At rest, the bottom pressure is $\pi_b = g\rho_N H$. When the ocean is moving this pressure becomes $\pi_b = g\rho_N D$. Because of this movement, the ocean bottom experiences a pressure deficit which is :

$$\delta\pi_b = -g\rho_N(D - H) \tag{65}$$

Since before we used the decomposition (1), the potential is therefore written :

$$\delta\pi_b = -g\eta\rho_N H \tag{66}$$

As the potential Π_N reflects the pressure deficit at the bottom in an ocean where the density evolves from ρ_1 in the surface layer to ρ_N in the bottom layer, to obtain the baroclinic Montgomery potential at the bottom we must also take into account the pressure deficit at the bottom which experiences a homogeneous ocean of instantaneous height $D = (1+\eta)H$ and of density ρ_1 :

$$\delta'\pi_b = -g\eta\rho_1 H \tag{67}$$

We finally obtain the general form giving the Montgomery potential at the bottom :

$$M_N = \Pi_N - g\eta H(\rho_N + \rho_1) \tag{68}$$

Once we calculate the potential M_N, we apply the form (63) to obtain the potential M_k of each layer up to the surface. In fact, in MICOM 2.6, the surface layer is a mixed layer where the density ρ_s varies with time. Now, in the preceding calculations, a constant value ρ_1 serves to represent the surface density. To express the Montgomery potential for the mixed layer, we need to account for this particularity. So, to free us from this problem, using the recurrence relation (63), we write :

$$M_1 = M_2 + g(\rho_2 - \rho_s)h_1 \tag{69}$$

Then :

$$M_1^{fin} = M_1 + h_1\left[(\rho_s - \rho_1) - (\rho_2 - \rho_1)\right] \tag{70}$$

For the other part, if we express the sum over the N layers of the decomposition (1) using the form : $p_b = p'_b + p''_b$ where :

$$p''_b = \eta \sum_{k=1}^{N} \Delta p'_k = \eta p'_b, \tag{71}$$

we can then calculate in time the new report of the perturbation η to the reference ocean by the relation :

$$\eta = \frac{p''_b}{p'_b} \tag{72}$$

4.1.2 Bottom drag

In MICOM 2.6, to model dissipation by bottom drag, we introduce the quadratic form :

$$\boldsymbol{\tau}_b = C_D \left|\overline{\mathbf{u}}_b\right| \overline{\mathbf{u}}_b \tag{73}$$

C_D is a drag coefficient. In version 2.6 of MICOM, $\overline{\mathbf{u}}_b$ represents the average velocity in a slice of water of thickness δz situated just above the bottom. We set $\delta z = 10\ m$. For the case when the thickness of layer N is less than the fixed value δz, it is necessary to use an average (the sum of the thicknesses of layers $N, N-1, \ldots, k$) :

$$\overline{\mathbf{u}'}_b = \frac{1}{\delta z}\left(\sum_{l=N}^{k} \mathbf{u}'_l h'_l + \mathbf{u}'_{k-1}\delta h'_{k-1}\right) \tag{74}$$

with :

$$\delta z = \sum_{l=N}^{k} h'_l + \delta h'_{k-1} \tag{75}$$

and we now put :

$$\overline{\mathbf{u}}_b = \overline{\mathbf{u}} + \overline{\mathbf{u}'}_b \tag{76}$$

So that the drag remains always significant, we introduce a residual velocity $\overline{c}$ and finally, we calculate the expression :

$$\mathcal{D}_{i,j} = C_D\left(\sqrt{\overline{u}_b^2 + \overline{v}_b^2} + \overline{c}\right) \tag{77}$$

In MICOM 2.6, $\overline{c} = 10\ cm/s$ and $C_D = 3 \times 10^{-3}$. We now calculate the vertical divergence of the bottom-induced stress :

$$\frac{\partial \boldsymbol{\tau}_b}{\partial p} = \begin{cases} \frac{\partial \tau_{b_x}}{\partial p} \\ \\ \frac{\partial \tau_{b_y}}{\partial p} \end{cases} \tag{78}$$

Only the momentum in layers with indices $N, N-1, \ldots, k$ situated in the δz slice of water feels the dissipation term. Moreover, we suppose that the drag stress varies linearly inside each layer. So, if the bottom layer thickness is strictly equal to δz :

$$\left(\frac{\partial \tau_{b_x}}{\partial p}\right)_{i,j} = \frac{1}{2} u_{i,j,N} \left(\mathcal{D}_{i,j} + \mathcal{D}_{i-1,j}\right) / \delta z \tag{79}$$

On the other hand, when the bottom layer thickness is greater than δz, we must account for the fact that only a fraction of layer N feels the dissipation. We then simply write :

$$\left(\frac{\partial \tau_{b_x}}{\partial p}\right)'_{i,j} = \left(\frac{\partial \tau_{b_x}}{\partial p}\right)_{i,j} \frac{\delta z}{\Delta p'_{i,j,N}} \tag{80}$$

If the dissipation layer is made up of more layers, we apply similar reasoning to distribute the dissipation over the N_f layers concerned.

4.1.3 Influence of the wind

In MICOM 2.6, to model the surface mechanical effect of the wind, we use the monthly climatological files of wind stress $\boldsymbol{\tau}_s$, interpolating the values during the simulation with the aid of the coefficients $w0, w1, w2, w3$.

4.2 Usage

In MICOM 2.6, the numerical calculation of forcing terms in the momentum equations is realized by the subprogram :

`subroutine momeq1` (argument list)

4.2.1 Order of operations

The first step consists of calculating the Montgomery potential at the bottom. For this we utilize the relation (68). The potential Π_{N+1} is calculated from the field of initial conditions (subroutine **inicon**). Once we have evaluated the coefficient $(1+\eta)$ with the help of expression (72), it is then possible to determine the Montgomery potential at the N interfaces above by the relation (63).

```
c
c --- --------------------
c --- hydrostatic equation
c --- --------------------
c
      do 13 j=1,jj1
      do 13 l=1,isp(j)
```

```
c
      do 8 k=1,kk
      do 8 i=ifp(j,l),ilp(j,l)
    8 p(i,j,k+1)=p(i,j,k)+dp(i,j,k+mm)
c
      do 80 i=ifp(j,l),ilp(j,l)
      util1(i,j)=1.+pbavg(i,j,m)/p(i,j,kk+1)
   80 montg(i,j,kk)=psikk(i,j)-pbavg(i,j,m)*(theta(kk)+thbase)*thref
c
      do 81 k=kk-1,2,-1
      do 81 i=ifp(j,l),ilp(j,l)
   81 montg(i,j,k)=montg(i,j,k+1)
     .             +p(i,j,k+1)*(theta(k+1)-theta(k))*thref*util1(i,j)
c
      do 13 i=ifp(j,l),ilp(j,l)
   13 montg(i,j,1)=montg(i,j,2)
     .             +p(i,j,2)*(theta(2)-thmix(i,j,m))*thref*util1(i,j)
c
```

Next, we estimate the invariant part (along the vertical) of the bottom drag by applying the formula (77) introduced in Section (4.1.2).

```
c
c --- bottom drag (standard bulk formula)
c
      do 804 j=1,jj1
      do 804 l=1,isp(j)
c
      do 800 i=ifp(j,l),ilp(j,l)
      util1(i,j)=0.
  800 util2(i,j)=0.
c
      do 801 k=1,kk
      kn=k+nn
      do 801 i=ifp(j,l),ilp(j,l)
      phi=max(p(i,j,k+1),p(i,j,kk+1)-tenm)
      plo=max(p(i,j,k ),p(i,j,kk+1)-tenm)
      util1(i,j)=util1(i,j)+(u(i,j,kn)+u(i+1,j,kn))*(phi-plo)
  801 util2(i,j)=util2(i,j)+(v(i,j,kn)+v(i,j+1,kn))*(phi-plo)
c
      do 804 i=ifp(j,l),ilp(j,l)
      ubot=ubavg(i,j,n)+ubavg(i+1,j,n)+util1(i,j)/tenm
      vbot=vbavg(i,j,n)+vbavg(i,j+1,n)+util2(i,j)/tenm
  804 drag(i,j)=.003*(.25*sqrt(ubot*ubot+vbot*vbot)+cbar)*g/tenm
c
```

For each of the two horizontal components, the following step does the following operations

in order :

1) Calculation of the spatial average of the barotropic vorticity originating from the barotropic-baroclinic splitting (*cf.* § 5, system 81).

2) Time interpolation of the wind stress.

3) Determination of the wind forcing of the surface layer term, described in Section 4.1.3 to which we adjoin the particular calculation of the Montgomery potential gradient for the mixed layer.

```
c
c --- store r.h.s.  of barotropic u/v eqn.  in -ubrhs,vbrhs-
c --- store layer 1 pressure gradient plus wind forcing in -gradx,grady-
c --- press.gradient is written in the form alpha * grad p + grad phi
c
      th2=thref*(1.-thbase-theta(2))
c
      do 69 j=1,jj1
      do 69 l=1,isu(j)
c
      do 89 i=ifu(j,l),ilu(j,l)
      ubrhs(i,j)=
     .  (vbavg(i ,j,m)*depthv(i ,j)+vbavg(i ,j+1,m)*depthv(i ,j+1)
     .  +vbavg(i-1,j,m)*depthv(i-1,j)+vbavg(i-1,j+1,m)*depthv(i-1,j+1))
     .  *(pvtrop(i,j)+pvtrop(i,j+1))*.125
c
      stresx=((taux(i,j,l0)+taux(i-1,j,l0))*w0
     .  +(taux(i,j,l1)+taux(i-1,j,l1))*w1
     .  +(taux(i,j,l2)+taux(i-1,j,l2))*w2
     .  +(taux(i,j,l3)+taux(i-1,j,l3))*w3)
     .  *g/(p(i,j,2)+p(i-1,j,2))
c
   89 gradx(i,j)=-stresx*scu(i) + .5*((p(i,j,2)-p(i-1,j,2))*thref*
     .  (1.-thbase-(thmix(i,j,m)*p(i,j,2)+thmix(i-1,j,m)*p(i-1,j,2))
     .  /( p(i,j,2) +p(i-1,j,2)))
     .  +(montg(i ,j,1)+montg(i ,j,2)-p(i ,j,2)*th2)
     .  -(montg(i-1,j,1)+montg(i-1,j,2)-p(i-1,j,2)*th2))
c
      do 69 k=1,kk
      do 69 i=ifu(j,l),ilu(j,l)
   69 pu(i,j,k+1)=pu(i,j,k)+dpu(i,j,k+mm)
c
      do 70 i=1,ii1
      do 70 l=1,jsv(i)
c
      do 90 j=jfv(i,l),jlv(i,l)
```

```
      vbrhs(i,j)=
     .-(ubavg(i,j ,m)*depthu(i,j )+ubavg(i+1,j ,m)*depthu(i+1,j )
     .  +ubavg(i,j-1,m)*depthu(i,j-1)+ubavg(i+1,j-1,m)*depthu(i+1,j-1))
     .  *(pvtrop(i,j)+pvtrop(i+1,j))*.125
c
      stresy=((tauy(i,j,l0)+tauy(i,j-1,l0))*w0
     .  +(tauy(i,j,l1)+tauy(i,j-1,l1))*w1
     .  +(tauy(i,j,l2)+tauy(i,j-1,l2))*w2
     .  +(tauy(i,j,l3)+tauy(i,j-1,l3))*w3)
     .  *g/(p(i,j,2)+p(i,j-1,2))
c
   90 grady(i,j)=-stresy*scv(i) + .5*((p(i,j,2)-p(i,j-1,2))*thref*
     .  (1.-thbase-(thmix(i,j,m)*p(i,j,2)+thmix(i,j-1,m)*p(i,j-1,2))
     .  /( p(i,j,2) +p(i,j-1,2)))
     .  +(montg(i,j ,1)+montg(i,j ,2)-p(i,j ,2)*th2)
     .  -(montg(i,j-1,1)+montg(i,j-1,2)-p(i,j-1,2)*th2))
c
      do 70 k=1,kk
      do 70 j=jfv(i,l),jlv(i,l)
   70 pv(i,j,k+1)=pv(i,j,k)+dpv(i,j,k+mm)
c
```

4.2.2 Flowchart

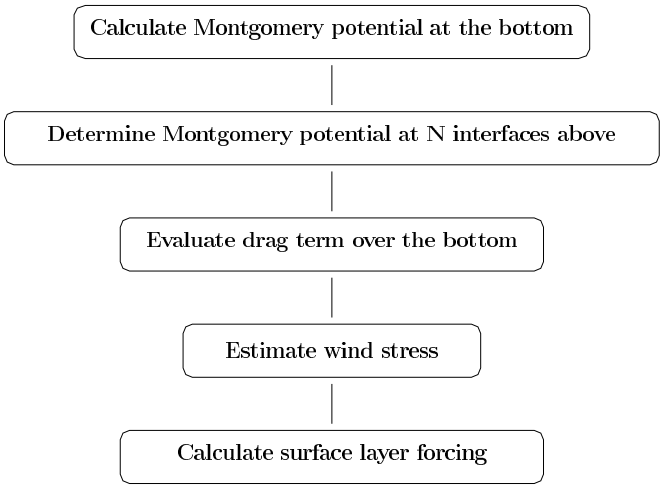

Figure 7: *Order of the treatment of the forcing terms in the momentum equations*

4.3 Variables

4.3.1 Identification

Notation in the theory	Notation in **momeq1.f**
$\mathcal{D}_{i,j}$	`drag(i,j)`
$M_{i,j,k}$	`montg(i,j,k)`
$p''_b = \eta p'_b$	`pbavg(i,j)`
$\overline{u}_b, \overline{v}_b$	`ubot(i,j),vbot(i,j)`
$\Pi_{i,j,N+1}$	`psikk(i,j)`
$\overline{u'}_b, \overline{v'}_b$	`util1(i,j),util2(i,j)`
δz	`tenm`
$\zeta\overline{u}, \zeta\overline{v}$	`ubrhs(i,j),vbrhs(i,j)`
ρ_s, ρ_k	`thmix(i,j,n), theta(k)`
τ_{s_x}, τ_{s_y}	`stresx,stresy`

4.3.2 Global variables

integer variables : `ifp(j,m)` `ilp(j,m)`, `ifv(j,m)`, `ilv(j,m)`, `isp(j)`, `isu(j)`, `isv(j)`, `kk`

real variables : `depthu(i,j)`, `depthv(i,j)`, `drag(i,j)`, `gradx(i,j)`, `grady(i,j)`, `depthu(i,j)`, `depthv(i,j)`, `dp(i,j,k)`, `dpu(i,j,k)`, `dpv(i,j,k)`, `montg(i,j,k)`, `p(i,j,k)`, `pbavg(i,j,n)`, `pu(i,j,k)`, `psikk(i,j)`, `pv(i,j,k)`, `pvtrop(i,j)`, `scu(i)`, `scv(i)`, `taux(i,j)`, `tauy(i,j)`, `thbase`, `thmix(i,j,n)`, `ubrhs(i,j)`, `util1(i,j)`, `util2(i,j)`, `vbrhs(i,j)`, `vflux(i,j)`, `vflx(i,j,k)`, `w0,w1,w2,w3`

4.3.3 Local Variables

`ia,ib,ja,jb`	intermediate indices
`phi,plo`	intermediate variables in the calculation of the average velocity near the bottom
`ubot,vbot`	components of velocity near the bottom
`stresx,stresy`	components of the wind stress
`th2`	deviation of density between the two first reference layers

5 Momentum : momeq2.f

To use the barotropic-baroclinic splitting of motion given by the form (2), once we have established the system (94) describing the behavior of the barotropic mode, by simple subtraction with the general system (60), we finally obtain the baroclinic system (BARAILLE & FILATOFF, 1995) :

$$\begin{aligned}
&\frac{\partial u'_k}{\partial t} + \frac{1}{2}\frac{\partial (u_k^2 + v_k^2)}{\partial x} - (\zeta_k + f)v'_i - \zeta_k \overline{v} + \frac{\partial}{\partial x}\left[M_k - \frac{1}{\rho_r}(\eta p'_b)\right] = -\frac{\partial \overline{u}^*}{\partial t} \\
&\frac{\partial v'_k}{\partial t} + \frac{1}{2}\frac{\partial (u_k^2 + v_k^2)}{\partial y} - (\zeta_k - f)u'_i - \zeta_k \overline{u} + \frac{\partial}{\partial y}\left[M_k - \frac{1}{\rho_r}(\eta p'_b)\right] = -\frac{\partial \overline{v}^*}{\partial t}
\end{aligned} \tag{81}$$

The index $_k$ represents the range of the isopycnic layer considered.

5.1 Formalism and numerical techniques

5.1.1 Numerical scheme

A leapfrog numerical scheme is used.

5.1.2 Turbulent viscosity

Though the nonlinear terms present in the momentum conservation equations of system (81) will not be called to play a significant role in ocean-scale applications, the equations are expressed in their complete form. The horizontal turbulent viscosity is defined by the relation :

$$\nu_u = \max\left\{u_d \Delta x, \lambda \left[\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2\right]^{1/2} \Delta x^2\right\} \tag{82}$$

u_d is a bottom diffusion velocity and λ a constant. In MICOM 2.6, we put : $u_d = 2\ cm/s$ and $\lambda = 1$.

5.1.3 Turbulent momentum flux

To the general equations given by the system (81), we add a diffusion term of the general form $(\Delta p)^{-1}\nabla \cdot (\nu_u \Delta p \nabla \mathbf{u})$. Similarly to the flux term $\nu_T \Delta p \nabla T$ appearing in the heat conservation equation (*cf.* § 3), the pressure increment Δp present in the momentum flux expression is replaced by the harmonic average $\widetilde{\Delta p} = 2/(\Delta p_{i-1}^{-1} + \Delta p_i^{-1})$ (*cf.* § 3.1.3). Moreover, writing this term in finite differences necessitates taking into account the possible presence of elevated bottom in neighboring cells. So, near partial boundaries, ($0 < w_{N,S,E,W} < 1$, (*cf.* § 5.1.5), the turbulent flux results from the sum :

1) of flux pertaining to the portion of the cell boundary facing bathymetry; this flux is subject to the boundary condition ($s_L \pm 1$) ;

2) of flux relating to the open part of the cell.

5.1.4 Intersection with the bathymetry

When an isocline intersects the bathymetry, a computational point of the Montgomery potential is then situated outside of the "wet" volume. In this case, in MICOM 2.6, the gradient of the Montgomery potential is calculated by the weighted average values of four points of the neighboring grid $(i',j') = (i \pm 1, j), (i, j \pm 1)$ (BLECK & SMITH, 1990 :

$$\left(\frac{\widetilde{\partial M}}{\partial x}\right)_{i,j,k} = \left[\sum_{i',j',k} \tilde{w}_{i',j',k} \left(\frac{\partial M}{\partial x}\right)_{i',j',k}\right] / \sum_{i',j'} \tilde{w}_{i',j',k} \tag{83}$$

The coefficients $\tilde{w}_{i',j',k}$ are defined by the formulae :

$$\begin{aligned} \tilde{w}_{i\pm1,j,k} &= \min\left(H_1, \Delta p'_{b_{i\pm1,j,k}}, \Delta p'_{s_{i\pm1,j,k}}\right) \\ \tilde{w}_{i,j\pm1,k} &= \min\left(H_1, \Delta p'_{b_{i,j\pm1,k}}, \Delta p'_{s_{i,j\pm1,k}}\right) \end{aligned} \tag{84}$$

with :

$$\Delta p'_{b_{i,j,k}} = p'_{b_{i,j}} - p'_{i,j,k} \quad \text{et} \quad \Delta p'_{s_{i,j,k}} = p'_{i,j,k+1} - p'_{s_{i,j}} \tag{85}$$

H_1 is the thickness given by weighting (83) of the Montgomery potential. For ocean applications, we typically have $H_1 = 10\ m$. Writing the coefficients $\tilde{w}_{i',j',k}$ allows us to simultaneously treat the problems of the intersection of interfaces with the bathymetry and that of the outcropping of interfaces with the surface. In fact, since MICOM 2.6 uses a surface mixed layer, it requires us to model the mechanism of outcropping of isopycnic layers into the mixed layer. We therefore put : $p'_{s_{i,j}} = p'_{i,j,2}$.
To avoid discontinuities, we introduce the following quantity :

$$\left(\frac{\partial M}{\partial x}\right)^{fin}_{i,j,k} = \left[\left(\frac{\partial M}{\partial x}\right)_{i,j,k} + (H_1 - \tilde{w}_{i,j,k}) \left(\frac{\widetilde{\partial M}}{\partial x}\right)_{i,j,k}\right] / H_1 \tag{86}$$

5.1.5 Boundary conditions

In the presence of a solid boundary, we introduce two types of physical boundary conditions :

1) slip conditon : $\partial u_s / \partial \pi = 0$;

2) no-slip conditon : $u_s = 0$.

u_s is the velocity component tangential to the boundary and π is the normal direction.
Suppose our domain is bounded by land to the East. In a point of (i,j,k) coordinates, to obtain an indication of the presence of this solid boundary, we calculate the quantity :

$$w_{E_{i,j.k}} = \max\left\{0, \min\left[1, \frac{p'_{i,j,k+1} - p'_{b_{i,j+1}}}{\max\left(p'_{i,j,k+1} - p'_{i,j,k}, \epsilon\right)}\right]\right\} \tag{87}$$

ϵ is used in the denominator of this expression to avoid division by zero. Three cases arise (*cf.* figure 9) :

1) $w_E = 1$ if the layer k intersects the bathymetry ;
2) $w_E = 0$ if it does not intersect the bathymetry ;
3) $0 < w_E < 1$ if only one part of the layer intersects the jump in the bathymetry existing between the grid points (i, j) and $(i, j + 1)$.

Introducing the variable s_L to represent the boundary condition chosen :

a) $s_L = 1$: slip ;
b) $s_L = -1$: no-slip

If a solid boundary is detected, it is then possible to express the particular behavior of the fluid in the immediate surroundings of the boundary by establishing the auxiliary velocity :

$$u_{E_{i,j,k}} = (1 - w_{E_{i,j,k}})u_{i,j,k} + w_{E_{i,j,k}} u_{i,j,k} s_L \tag{88}$$

So if an Eastern boundary is detected (w_E=1), we have :

a) if $s_L = 1$: $u_{E_{i,j,k}} = u_{i,j,k}$;
b) if $s_L = -1$: $u_{E_{i,j,k}} = -u_{i,j,k}$

For each layer k and at each point (i, j), we diagnose in this manner the presence of boundaries respectively to the North, South, East and then West of the domain by the calculation of the four coefficients w_N, w_S, w_E, w_W which permit us to establish the auxiliary velocity field u_N, u_S, u_E, u_W. This procedure remains valid as well for the case of a partial boundary $(0 < w_E < 1)$.

5.1.6 Vorticity

In the formalism of MICOM, the nonlinear terms and the Coriolis term are grouped to make the vorticity appear as ($\zeta = \partial v/\partial x - \partial u/\partial y$) :

$$\mathbf{u} \cdot \nabla \mathbf{u} + f\mathbf{k} \times \mathbf{u} = \nabla \frac{\mathbf{u}^2}{2} + (\zeta + f)\mathbf{k} \times \mathbf{u} \tag{89}$$

To conserve entropy, the components of the vorticity term $(\zeta + f)\mathbf{k} \times \mathbf{u}$ are written in the difference forms :

$$\begin{aligned} -v(\zeta + f) &= -\overline{V}^{x,y}\overline{Q}^{y} \\ u(\zeta + f) &= -\overline{U}^{x,y}\overline{Q}^{x} \end{aligned} \tag{90}$$

where we introduce the mass flux $(U, V) = u\Delta\overline{p}^{x}, v\Delta\overline{p}^{y}$ and the potential vorticity $Q = (\zeta + f)/\Delta\overline{p}^{x,y}$. The operator $\overline{(\ )}$ permits the recentering of differences of pressure at computational points of the velocity components on a C grid. In an isopycnic shallow-water model, there is no guarantee that the instantaneous thickness of the considered layer remains non-zero. To address this problem, (Bleck & Smith 1990, p 3283) developed a particular method. It is called the "one-eighths rule" and remedies abnormal situations which arise when certain layer thickness values vanish. So, before estimating the potential vorticity field $Q_{i,j,k}$, we calculate for each layer, at each point, the sum of two larger values of the variable Δp among the 4 points

surrounding the point considered. We denote this sum : $\Pi_{i,j}$. The denominator which we use in the vorticity expression then takes the value :

$$\Delta p_{max} = \max\left\{\Delta\overline{p}^{x,y}_{i,j}, \frac{1}{8}\max\left(\Pi_{i,j}, \Pi_{i-1,j}, \Pi_{i+1,j}, \Pi_{i,j-1}, \Pi_{i,j+1}\right)\right\} \tag{91}$$

In the case where the two arguments of this collation are zero, we introduce a residual numerical value δp. In MICOM 2.6, we use $\delta p = 10^{-2}\ m$.

5.2 Usage

In MICOM 2.6, the numerical evolution of the baroclinic velocity components is realized by the subprogram :

`subroutine momeq2` (argument list)

5.2.1 Order of operations

The algorithm follows a treatment by layer.

The first step consists of calculating the total currents at times $(n-1)\Delta t$ and $n\Delta t$ and the associated fluxes. Then, to determine the optimum value of the difference in pressure given by the formula (91).

```
c
c --- store total (barotropic plus baroclinic) flow at old and mid time in
c --- -utotn,vtotn- and -utotm,vtotm- respectively.  store minimum thickness
c --- values for use in pot.vort.  calculation in -dpmx-.
c
      do 802 j=1,jj1
      do 802 l=1,isu(j)
      do 802 i=ifu(j,l),ilu(j,l)
      dpmx(i,j )=max(dpmx(i,j ),dp(i,j,km)+dp(i-1,j,km))
      uflux(i,j)=(u(i,j,km)+ubavg(i,j,m))*max(dpu(i,j,km),onecm)
      utotm(i,j)=u(i,j,km)+ubavg(i,j,m)
  802 utotn(i,j)=u(i,j,kn)+ubavg(i,j,n)
c
      do 807 j=1,jj1
      do 807 l=1,isu(j)
      do 807 i=ifu(j,l),ilu(j,l)
  807 dpmx(i,j+1)=max(dpmx(i,j+1),dp(i,j,km)+dp(i-1,j,km))
c
      do 803 i=1,ii1
      do 803 l=1,jsv(i)
      do 803 j=jfv(i,l),jlv(i,l)
      dpmx(i ,j)=max(dpmx(i ,j),dp(i,j,km)+dp(i,j-1,km))
      vflux(i,j)=(v(i,j,km)+vbavg(i,j,m))*max(dpv(i,j,km),onecm)
      vtotm(i,j)=v(i,j,km)+vbavg(i,j,m)
  803 vtotn(i,j)=v(i,j,kn)+vbavg(i,j,n)
```

```
c
      do 808 i=1,ii1
      do 808 l=1,jsv(i)
      do 808 j=jfv(i,l),jlv(i,l)
  808 dpmx(i+1,j)=max(dpmx(i+1,j),dp(i,j,km)+dp(i,j-1,km))
c
```

The next operation consists of calculating indicators of the presence of bottom topography in neighboring cells by the method described in Section (5.1.5).

```
c
c --- define auxiliary velocity fields (via,vib,uja,ujb) to implement
c --- sidewall friction along near-vertical bottom slopes.  wgtja,wgtjb,wgtia,
c --- wgtib indicate the extent to which a sidewall is present.
c
      do 885 j=1,jj1
c --- if j=1, j-1 must point to zero-filled row (same for j+1 in case j=jj1)
      ja=mod(j-2+jj,jj)+1
      jb=j+1
      do 805 l=1,isu(j)
      do 805 i=ifu(j,l),ilu(j,l)
      wgtja(i,j)=max(0.,min(1.,(pu(i,j,k+1)-depthu(i,ja))
     .  /max(pu(i,j,k+1)-pu(i,j,k),epsil)))
      wgtjb(i,j)=max(0.,min(1.,(pu(i,j,k+1)-depthu(i,jb))
     .  /max(pu(i,j,k+1)-pu(i,j,k),epsil)))
      uja(i,j)=(1.-wgtja(i,j))*utotn(i,ja)+wgtja(i,j)*slip*utotn(i,j)
  805 ujb(i,j)=(1.-wgtjb(i,j))*utotn(i,jb)+wgtjb(i,j)*slip*utotn(i,j)
c
c --- vorticity, pot.vort., defor.  at lateral boundary points
      do 885 l=1,isv(j)
      i=ifv(j,l)
      vort(i ,j)= vtotm(i,j)*(1.-slip)*scui(i)
      potvor(i ,j)=(vort(i ,j)+corio(i ,j))
     ./(max(4.*(dp(i,j,km)+dp(i,ja ,km)),dpmx(i,j),dpmx(i+1,j))*.125)
      defor2(i ,j)=(vtotn(i,j)*(1.-slip))**2
      i=ilv(j,l)
      vort(i+1,j)=-vtotm(i,j)*(1.-slip)*scui(i)
      potvor(i+1,j)=(vort(i+1,j)+corio(i+1,j))
     ./(max(4.*(dp(i,j,km)+dp(i,ja ,km)),dpmx(i,j),dpmx(i+1,j))*.125)
  885 defor2(i+1,j)=(vtotn(i,j)*(1.-slip))**2
c
      do 886 i=1,ii1
c --- if i=1, i-1 must point to zero-filled row (same for i+1 in case i=ii1)
      ia=mod(i-2+ii,ii)+1
      ib=i+1
      do 806 l=1,jsv(i)
      do 806 j=jfv(i,l),jlv(i,l)
```

```
      wgtia(i,j)=max(0.,min(1.,(pv(i,j,k+1)-depthv(ia,j))
     .  /max(pv(i,j,k+1)-pv(i,j,k),epsil)))
      wgtib(i,j)=max(0.,min(1.,(pv(i,j,k+1)-depthv(ib,j))
     .  /max(pv(i,j,k+1)-pv(i,j,k),epsil)))
      via(i,j)=(1.-wgtia(i,j))*vtotn(ia,j)+wgtia(i,j)*slip*vtotn(i,j)
  806 vib(i,j)=(1.-wgtib(i,j))*vtotn(ib,j)+wgtib(i,j)*slip*vtotn(i,j)
c
c --- vorticity, pot.vort., defor.  at lateral boundary points
      do 886 l=1,jsu(i)
      j=jfu(i,l)
      vort(i,j )=-utotm(i,j)*(1.-slip)*scui(i)
      potvor(i,j )=(vort(i,j )+corio(i,j ))
     ./(max(4.*(dp(i,j,km)+dp(ia ,j,km)),dpmx(i,j),dpmx(i,j+1))*.125)
      defor2(i,j )=(utotn(i,j)*(1.-slip))**2
      j=jlu(i,l)
      vort(i,j+1)= utotm(i,j)*(1.-slip)*scui(i)
      potvor(i,j+1)=(vort(i,j+1)+corio(i,j+1))
     ./(max(4.*(dp(i,j,km)+dp(ia ,j,km)),dpmx(i,j),dpmx(i,j+1))*.125)
  886 defor2(i,j+1)=(utotn(i,j)*(1.-slip))**2
c
      do 63 j=1,jj1
      do 63 l=1,isp(j)
      do 63 i=ifp(j,l),ilp(j,l)
   63 defor1(i,j)=((utotn(i+1,j)-utotn(i,j))
     .  -(vtotn(i,j+1)-vtotn(i,j)))**2
c
c --- vorticity, pot.vort., defor.  at interior points (incl.  promontories)
      do 64 j=2,jj1
      do 64 l=1,isq(j)
      do 64 i=ifq(j,l),ilq(j,l)
      vort(i,j)=(vtotm(i,j)*scv(i)-vtotm(i-1,j)*scv(i-1)
     .  -utotm(i,j)*scu(i)+utotm(i,j-1)*scu(i ))*scu2i(i)
      potvor(i,j)=(vort(i,j)+corio(i,j))
     ./(max(2.*(dp(i,j,km)+dp(i-1,j,km)+dp(i,j-1,km)+dp(i-1,j-1,km))
     .,dpmx(i,j),dpmx(i-1,j),dpmx(i+1,j),dpmx(i,j-1),dpmx(i,j+1))*.125)
   64 defor2(i,j)=(vtotn(i,j)-via(i,j)
     .  +utotn(i,j)-uja(i,j))**2
c
```

In the next phase, we apply the relation (82) and we calculate the turbulent momentum fluxes by the procedure mentioned in Section (5.1.3). Then comes the inference of the weighted gradient of the Montgomery potential following the theory outlined in Section (5.1.4). If the layer concerned includes drag dissipation, it is integrated following the steps described in Section (4.1.2). We are then ready to proceed to the estimation of the new values of the velocity components.

```
c --- ----------
c --- u equation
```

```
c --- ----------
c
c --- deformation-dependent eddy viscosity coefficient
c
      do 37 j=1,jj1
      do 37 l=1,isu(j)
c
      do 47 i=ifu(j,l),ilu(j,l)
   47 visc(i,j)=scu(i)*max(veldff,viscos*
     .sqrt(.5*(defor1(i,j)+defor1(i-1,j)+defor2(i,j)+defor2(i,j+1))))
c
      visc(ifu(j,l)-1,j)=visc(ifu(j,l),j)
   37 visc(ilu(j,l)+1,j)=visc(ilu(j,l),j)
c
      do 820 j=1,jj1
c --- if j=1, j-1 must point to zero-filled row (same for j+1 in case j=jj1)
      ja=mod(j-2+jj,jj)+1
      jb=j+1
      do 820 l=1,isu(j)
c
c --- longitudinal turb.  momentum flux (at mass points)
c
      do 824 i=ifu(j,l)-1,ilu(j,l)
  824 uflux1(i,j)=.5*(visc(i,j)+visc(i+1,j))*(utotn(i,j)-utotn(i+1,j))
     .  *harmon(max(dpu(i ,j,km),onemm),
     .  max(dpu(i+1,j,km),onemm))
c
c --- lateral turb.  momentum flux (at vorticity points)
c --- (left and right fluxes are evaluated separately because of sidewalls)
c
      do 820 i=ifu(j,l),ilu(j,l)
      dpij=max(dpu(i,j ,km),onemm)
      dpja=max(dpu(i,ja,km),onemm)
      dpjb=max(dpu(i,jb,km),onemm)
      if (iu(i,ja).eq.0) then
      viscja=visc(i,j )
      else
      viscja=visc(i,ja)
      end if
      if (iu(i,jb).eq.0) then
      viscjb=visc(i,j )
      else
      viscjb=visc(i,jb)
      end if
      uflux2(i,j)=.5*(visc(i,j)+viscja)*(uja(i,j)-utotn(i,j))
     .  *harmon(dpja+wgtja(i,j)*(dpij-dpja),dpij)
      uflux3(i,j)=.5*(visc(i,j)+viscjb)*(utotn(i,j)-ujb(i,j))
```

```
     .  *harmon(dpjb+wgtjb(i,j)*(dpij-dpjb),dpij)
c
c --- spatial weighting function for pressure gradient calculation:
      util1(i,j)=max(0.,min(depthu(i,j)-pu(i,j,k),
     .  pu(i,j,k+1)-pu(i,j,2),h1))
  820 pgfx(i,j)=(montg(i,j,k)-montg(i-1,j,k))*util1(i,j)
c
      do 6 j=1,jj1
c --- if j=1, j-1 must point to zero-filled row (same for j+1 in case j=jj1)
      ja=mod(j-2+jj,jj)+1
      jb=j+1
      do 6 l=1,isu(j)
c
      if (k.gt.1) then
c
c --- pressure force in x direction
c --- ('scheme 2' from appendix -a- in bleck-smith paper)
c
      do 98 i=ifu(j,l),ilu(j,l)
   98 gradx(i,j)=(pgfx(i,j)+(h1-util1(i,j))*
     .  (pgfx (i-1,j)+pgfx (i+1,j)+pgfx (i,ja )+pgfx (i,jb ))/
     .  (util1(i-1,j)+util1(i+1,j)+util1(i,ja )+util1(i,jb )+epsil))/h1
      endif
c
      do 6 i=ifu(j,l),ilu(j,l)
c
      ptopl=min(depthu(i,j),.5*(p(i,j,k )+p(i-1,j,k )))
      pbotl=min(depthu(i,j),.5*(p(i,j,k+1)+p(i-1,j,k+1)))
c
c --- bottom boundary layer stress
      botstr(i,j)=-utotn(i,j)*.5*(drag(i,j)+drag(i-1,j))*
     .  (max(depthu(i,j)-tenm, pbotl)
     .  -max(depthu(i,j)-tenm,min(ptopl,pbotl-tenm)))
     .  /max(dpu(i,j,km),tenm)
c
      uold(i,j,k)=u(i,j,kn)
    6 u(i,j,kn)=u(i,j,kn)+delt1*(-gradx(i,j)*scui(i)
     .-.25*(utotm(i+1,j)**2+vtotm(i ,j)**2+vtotm(i ,j+1)**2
     .  -utotm(i-1,j)**2-vtotm(i-1,j)**2-vtotm(i-1,j+1)**2)*scui(i)
     .+.125*(vflux(i ,j)+vflux(i ,j+1)+vflux(i-1,j)+vflux(i-1,j+1))
     .  *(potvor(i,j)+potvor(i,j+1)) - ubrhs(i,j) + botstr(i,j)
     .-(uflux1(i,j)-uflux1(i-1,j)
     .  +uflux3(i,j)-uflux2(i,j))/(scu2(i)*max(dpu(i,j,km),onemm)))
c
c --- ----------
c --- v equation
c --- ----------
```

```
c
c --- deformation-dependent eddy viscosity coefficient
c
      do 38 i=1,ii1
      do 38 l=1,jsv(i)
c
      do 48 j=jfv(i,l),jlv(i,l)
   48 visc(i,j)=scv(i)*max(veldff,viscos*
     .sqrt(.5*(defor1(i,j)+defor1(i,j-1)+defor2(i,j)+defor2(i+1,j))))
c
      visc(i,jfv(i,l)-1)=visc(i,jfv(i,l))
   38 visc(i,jlv(i,l)+1)=visc(i,jlv(i,l))
c
      do 821 i=1,ii1
c --- if i=1, i-1 must point to zero-filled row (same for i+1 in case i=ii1)
      ia=mod(i-2+ii,ii)+1
      ib=i+1
      do 821 l=1,jsv(i)
c
c --- longitudinal turb.  momentum flux (at mass points)
c
      do 826 j=jfv(i,l)-1,jlv(i,l)
  826 vflux1(i,j)=.5*(visc(i,j)+visc(i,j+1))*(vtotn(i,j)-vtotn(i,j+1))
     .  *harmon(max(dpv(i,j ,km),onemm),
     .  max(dpv(i,j+1,km),onemm))
c
c --- lateral turb.  momentum flux (at vorticity points)
c --- (left and right fluxes are evaluated separately because of sidewalls)
c
      do 821 j=jfv(i,l),jlv(i,l)
      dpij=max(dpv(i ,j,km),onemm)
      dpia=max(dpv(ia,j,km),onemm)
      dpib=max(dpv(ib,j,km),onemm)
      if (iv(ia,j).eq.0) then
      viscia=visc(i ,j)
      else
      viscia=visc(ia,j)
      end if
      if (iv(ib,j).eq.0) then
      viscib=visc(i ,j)
      else
      viscib=visc(ib,j)
      end if
      vflux2(i,j)=.5*(visc(i,j)+viscia)*(via(i,j)-vtotn(i,j))
     .  *harmon(dpia+wgtia(i,j)*(dpij-dpia),dpij)
      vflux3(i,j)=.5*(visc(i,j)+viscib)*(vtotn(i,j)-vib(i,j))
     .  *harmon(dpib+wgtib(i,j)*(dpij-dpib),dpij)
```

```
c
c --- spatial weighting function for pressure gradient calculation:
      util2(i,j)=max(0.,min(depthv(i,j)-pv(i,j,k),
     .  pv(i,j,k+1)-pv(i,j,2),h1))
  821 pgfy(i,j)=(montg(i,j,k)-montg(i,j-1,k))*util2(i,j)
c
      do 7 i=1,ii1
c --- if i=1, i-1 must point to zero-filled row (same for i+1 in case i=ii1)
      ia=mod(i-2+ii,ii)+1
      ib=i+1
      do 7 l=1,jsv(i)
c
      if (k.gt.1) then
c
c --- pressure force in y direction
c --- ('scheme 2' from appendix -a- in bleck-smith paper)
c
      do 99 j=jfv(i,l),jlv(i,l)
   99 grady(i,j)=(pgfy(i,j)+(h1-util2(i,j))*
     .  (pgfy (ia ,j)+pgfy (ib ,j)+pgfy (i,j-1)+pgfy (i,j+1))/
     .  (util2(ia ,j)+util2(ib ,j)+util2(i,j-1)+util2(i,j+1)+epsil))/h1
      endif
c
      do 7 j=jfv(i,l),jlv(i,l)
c
      ptopl=min(depthv(i,j),.5*(p(i,j,k )+p(i,j-1,k )))
      pbotl=min(depthv(i,j),.5*(p(i,j,k+1)+p(i,j-1,k+1)))
c
c --- bottom boundary layer stress
      botstr(i,j)=-vtotn(i,j)*.5*(drag(i,j)+drag(i,j-1))*
     .  (max(depthv(i,j)-tenm, pbotl)
     .  -max(depthv(i,j)-tenm,min(ptopl,pbotl-tenm)))
     .  /max(dpv(i,j,km),tenm)
c
      vold(i,j,k)=v(i,j,kn)
    7 v(i,j,kn)=v(i,j,kn)+delt1*(-grady(i,j)*scvi(i)
     .-.25*(vtotm(i,j+1)**2+utotm(i,j )**2+utotm(i+1,j )**2
     .  -vtotm(i,j-1)**2-utotm(i,j-1)**2-utotm(i+1,j-1)**2)*scvi(i)
     .-.125*(uflux(i,j )+uflux(i+1,j )+uflux(i,j-1)+uflux(i+1,j-1))
     .  *(potvor(i,j)+potvor(i+1,j)) - vbrhs(i,j) + botstr(i,j)
     .-(vflux1(i,j)-vflux1(i,j-1)
     .  +vflux3(i,j)-vflux2(i,j))/(scv2(i)*max(dpv(i,j,km),onemm)))
c
```

5.2.2 Flowchart

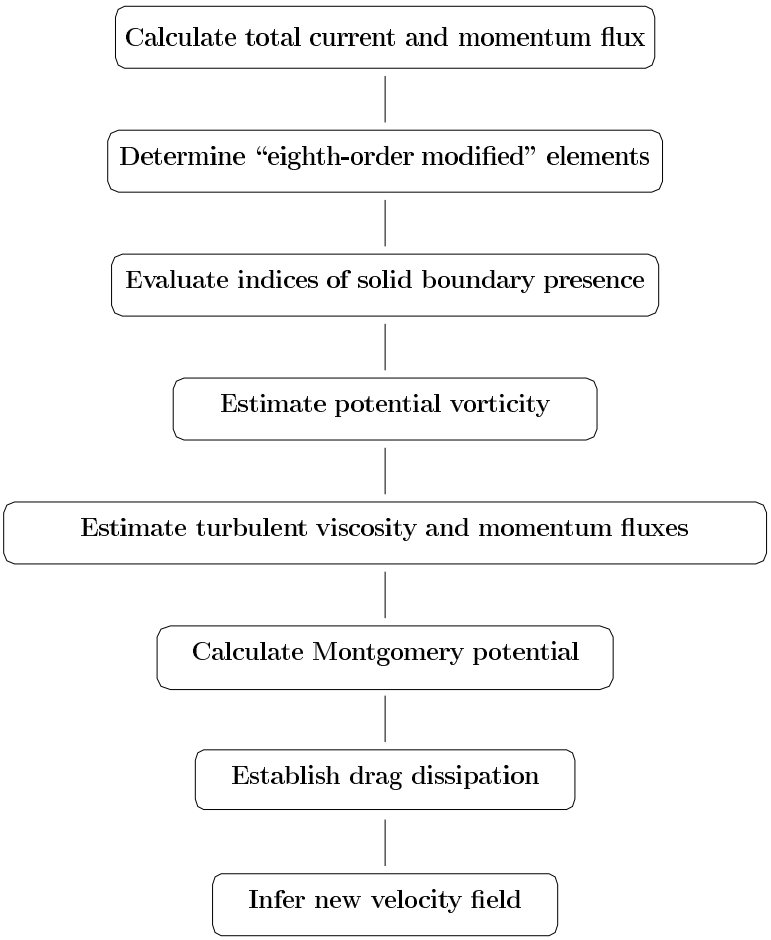

Figure 8: *Order of the treatment of the momentum equation*

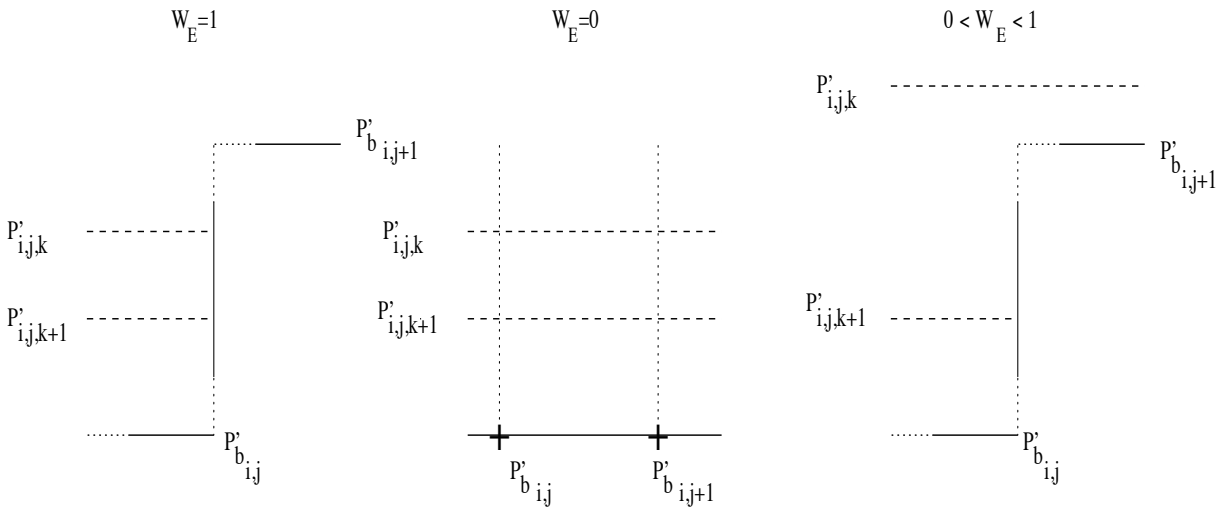

Figure 9: *Schematic of the intersection of layers with solid boundaries*

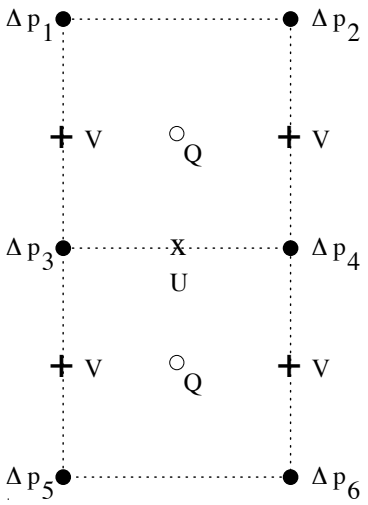

Figure 10: *Distribution of variables used in the evaluation of the Coriolis term at the central point* $u(i,j)$

5.3 Variables

5.3.1 Identification

Notation in the theory	Notation in **momeq2.f**
$f(x,y)$	`corio(i,j)`
H_1	`h1`
$(\partial M/\partial x)_{i',j'}, (\partial M/\partial y)_{i',j'}$	`pgfx(i,j),pgfy(i,j)`
$(\partial M/\partial x)_{i,j}^{fin}, (\partial M/\partial y)_{i,j}^{fin}$	`gradx(i,j),grady(i,j)`
$\Pi_{i,j}$	`dpmx(i,j)`
$(u_x - v_y)^2$	`defor1(i,j)`
$(v_x + u_y)^2$	`defor2(i,j)`
u_d	`veldff`
u_O, u_E	`uja(i,j),ujb(i,j)`
v_N, v_S	`via(i,j),vib(i,j)`
w_O, w_E	`wgtja(i,j),wgtjb(i,j)`
w_N, w_S	`wgtia(i,j),wgtib(i,j)`
$\tilde{w}_{i',j'}$	`util1(i,j)`

$\zeta(x,y)$	vort(i,j)
λ	viscos
ν	visc(i,j)
ϵ	epsil

5.3.2 Global variables

integer variables : ii, ii1, ifp(j,m), ifq(j,m), ilp(j,m), ilq(j,m), ifv(j,m), ilv(j,m), isp(j), isq(j), isu(j), isv(j), iu(i,j), iv(i,j), jfu(i,l), jfv(i,l), jlv(i,l), jlu(i,l), jsu(i), jsv(j), jj, jj1, kk

real variables : corio(i,j), defor1(i,j), defor2(i,j), depthu(i,j), depthv(i,j), dpmx(i,j), dp(i,j,k), dpu(i,j,k), dpv(i,j,k), drag(i,j), gradx(i,j), grady(i,j), montg(i,j,k), onecm, p(i,j,k), pgfx(i,j), pgfy(i,j), potvor(i,j), pu(i,j,k), pv(i,j,k), scu(i), scv(i), slip, ubrhs(i,j), uja, ujb, uold(i,j,k), util1(i,j), util2(i,j), utotm(i,j), utotn(i,j), vbrhs(i,j), uflux(i,j), uflux1(i,j), uflux2(i,j), uflux3(i,j), vflux(i,j), vflux1(i,j), vflux2(i,j), vflux3(i,j), visc(i,j), vold(i,j,k), vort(i,j), vtotm(i,j), vtotn(i,j) wgtia(i,j), wgtib(i,j), wgtja(i,j), wgtjb(i,j)

5.3.3 Local variables

botstr(i,j)	drag dissipation
ia,ib,ja,jb	intermediate indices
dpij,dpja,dpjb	intermediate variables in the calculation of turbulent flux in the presence of a boundary
ptopl,pbotl	intermediate variables in the drag calculation
ubot,vbot	velocity components near the bottom
viscja,viscjb	intermediate variables in the calculation of turbulent flux in the presence of a boundary

5.4 Procedures

Functions harmon

6 Barotropic mode : barotp.f

6.1 Formalism and numerical techniques

In introducing the decomposition (1) and summing over all layers of the general continuity equation, we obtain the form :

$$\frac{\partial \eta p'_b}{\partial t} + \nabla \cdot [(1+\eta)\overline{\mathbf{u}} p'_b] = 0 \tag{92}$$

To establish the corresponding equations of motion, we consider the average of the general form (60) in the sense of (3). Next we introduce the decomposition (2), then the relations (4) and (1). Finally, we obtain the system (Baraille & Filatoff, 1995) :

$$\begin{aligned} &\frac{\partial}{\partial t}\left[\overline{u}(1+\eta)\right] - (1+\eta) f \overline{v} + \frac{1}{\rho_r}\frac{\partial}{\partial x}\left[\eta p'_b (1+\eta)\right] + \frac{R + Q(\eta)}{p'_b} = 0 \\ &\frac{\partial}{\partial t}\left[\overline{v}(1+\eta)\right] + (1+\eta) f \overline{u} + \frac{1}{\rho_r}\frac{\partial}{\partial y}\left[\eta p'_b (1+\eta)\right] + \frac{R' + Q'(\eta)}{p'_b} = 0 \end{aligned} \tag{93}$$

which we write more simply :

$$\begin{aligned} &\frac{\partial \overline{u}}{\partial t} - f\overline{v} + \frac{1}{\rho_r}\frac{\partial}{\partial x}\left(\eta p'_b\right) = \frac{\partial \overline{u}^*}{\partial t} \\ &\frac{\partial \overline{v}}{\partial t} + f\overline{u} + \frac{1}{\rho_r}\frac{\partial}{\partial y}\left(\eta p'_b\right) = \frac{\partial \overline{v}^*}{\partial t} \end{aligned} \tag{94}$$

$Q(\eta)$ and $Q'(\eta)$ are expressions grouping the nonlinear terms. R and R' represent the components of pressure gradient induced by the stratification. The pseudo-velocity $\overline{\mathbf{u}}^*(\overline{u}^*, \overline{v}^*)$ assures the property (4).

We denote Δt_B, the time step of the external mode (barotropic) and Δt_b, the time step of the internal mode (baroclinic), such that :

$$\Delta t_b = \mathcal{N} \Delta t_B \tag{95}$$

The barotropic equations are therefore solved $\mathcal{N}$ times between two solutions of the baroclinic equations.

6.1.1 Rescaling of variables

As the variables are distributed on a C grid, a tendency calculation at velocity points which uses layer thicknesses obtained at height points by the application of formula (33) in the last step of the continuity equation necessitates a rescaling of the variables. In fact, during initialization (subroutine `inicon`), the depths at velocity points have been introduced by the relations :

$${}_u p'_{b_{i,j}} = \min\left(p'_{b_{i-1,j}}, p'_{b_{i,j}}\right) \quad \text{and} \quad {}_v p'_{b_{i,j}} = \min\left(p'_{b_{i,j-1}}, p'_{b_{i,j}}\right) \tag{96}$$

To retain consistency with the initial definitions, at the point of the u component, we introduce the average thickness :

$$\begin{aligned} {}_u\Delta p'_{i,j,k} &= \max\{0., \\ &\quad \min\left[{}_up'_{b_{i,j}}, 1/2\left({}_up'_{i,j,k+1} + {}_up'_{i-1,j,k+1}\right)\right] \\ &- \min\left[{}_up'_{b_{i,j}}, 1/2\left({}_up'_{i,j,k} + {}_up'_{i-1,j,k}\right)\right]\Big\} \end{aligned} \tag{97}$$

We use the equivalent formula at the point of the v component.

6.1.2 Rearrangement of the velocity profile

When the thickness of a layer is so small as to be considered numerically zero, it may still contain momentum. When layer k "disappears", we consider that, in terms of momentum, it still exists and acquires the momentum of the layer above. To translate this mechanism, we introduce the variable q :

$$q = \delta - \min\left({}_u\Delta p'_k, \delta\right) = \begin{cases} 0 & \text{if } {}_u\Delta p'_k \geq \delta \\ \delta - {}_u\Delta p'_k & \text{if } {}_u\Delta p'_k < \delta, \end{cases} \tag{98}$$

which permits us to define the weighted value (ditto for the v component) :

$$\underline{u}_k = \frac{1}{\delta}\left[u'_k\left({}_u\Delta p'_k\right) + u'_{k-1}(\delta - {}_u\Delta p'_k)\right] \tag{99}$$

In MICOM 2.6, we set $\delta = 10^{-1}\ m$.

6.1.3 Filtering

In the same manner as in the advection step (*cf.* § 3), to fix the problems of dispersion caused by the leapfrog scheme, we introduce Asselin filtering for the velocity. Take the component u' of the baroclinic velocity for the layer k and the node (i, j). We then write

$$\widehat{u'}^n\widehat{\Delta p'}^n = \left[u'^n(1-2\gamma)(\Delta p'^n + \epsilon) + \gamma\left(\widehat{u'}^{n-1}\widehat{\Delta p'}^{n-1} + u'^{n+1}\Delta p'^{n+1}\right)\right] \tag{100}$$

We introduce a residual thickness ϵ for which this form remains valid when ${}_u\Delta p'_{i,j,k} \to 0$. In MICOM, ϵ is set to a numerical value of $10^{-3}\ m$. We have $\gamma = 0.25$. The thickness of the layer ${}_u\Delta p'_{i,j,k}$ is also filtered by the formula :

$$\widehat{\Delta p'}^n = (1-2\gamma)(\Delta p'^n + \epsilon) + \gamma\left(\widehat{\Delta p'}^{n-1} + \Delta p^{n+1}\right) \tag{101}$$

6.1.4 Continuity equation

The continuity equation (92) is treated with the simplification $(1+\eta) \approx 1$. As we already indicated in Section 2, using this approximation does not perturb the property : $\partial p'_b/\partial t = 0$ (Baraille & Filatoff, 1995). The treated variable is therefore $p''_b = \eta p'_b$. Combining forward time-stepping

$$P''^{m+1}_b = P''^m_b - \Delta t_B\ \nabla\cdot\left(\overline{\mathbf{u}}p'_b\right)^m \tag{102}$$

with Asselin time filtering we arrive at

$$P_b''^{m+1} = (1 - \mathrm{w})P_b''^{m} + \mathrm{w}P_b''^{m-1} - \Delta t_B(1 + \mathrm{w})\nabla \cdot (\overline{\mathbf{u}}p_b')^m \tag{103}$$

and we set : $\mathrm{w} = 0.125$

6.1.5 Equations of motion

The equations of motion given by the system (94) call the reference density ρ_r introduced to represent the ocean of reference depth H. In MICOM 2.6, we make the identification $\rho_r \equiv \rho_0$ (*cf.* § 12).

The vector $\partial\overline{\mathbf{u}}^*/\partial t$ appearing in the right hand side of the equations of system (94) can then be seen as a forcing term in the generation of the linear barotropic mode. The solution of this system necessitates therefore the extraction of the component $\overline{\mathbf{u}}^*$. In the preceding step (*subroutine* `momeq2`), we calculate the baroclinic velocity profile expressed by the variable $\mathbf{u}_k'' = \mathbf{u}_k' + \overline{\mathbf{u}}^*$. In carrying out the sums :

$$S_u = \sum_{k=1}^{N} u_k''\Delta p_k' \quad \text{and} \quad S_v = \sum_{k=1}^{N} v_k''\Delta p_k' \tag{104}$$

and in accounting for the property (4), we therefore see that :

$$\overline{u}^* = \frac{S_u}{p_b'} \quad \text{and} \quad \overline{v}^* = \frac{S_v}{p_b'} \tag{105}$$

We note that the pseudo-vector $\overline{\mathbf{u}}^*$ is not a variable of state in the system whose evolution we seek. This is to say that in the preceding step (subroutine `momeq2`), we effectively inferred the transition : $\mathbf{u}_{i,j,k}'^{n} \rightarrow (\mathbf{u}_k' + \overline{\mathbf{u}}^*)_{i,j}^{n+1}$. Moreover, we also introduce the weighting w such that :

$$\overline{u}_{i,j}^{m+1} = (1 - \mathrm{w})\overline{u}_{i,j}^{m} + \mathrm{w}\overline{u}_{i,j}^{m-1} - \Delta t_B(1 + \mathrm{w})\left[-\alpha_0\left(\frac{\partial p_b''}{\partial x}\right)_{i,j}^{m+1} + \overline{f\overline{v}}_{i,j}^{m} + \overline{u}_{i,j}^{*,n+1}/\Delta t_b\right] \tag{106}$$

The Coriolis term is expressed by the centered form :

$$\overline{f\overline{v}}_{i,j} = 1/8(f_{i,j}' + f_{i,j+1}')\left[(\overline{v}_v p_b')_{i,j} + (\overline{v}_v p_b')_{i-1,j} + (\overline{v}_v p_b')_{i,j+1} + (\overline{v}_v p_b')_{i-1,j+1}\right] \tag{107}$$

with the barotropic potential vorticity f' defined as

$$f_{i,j}' = \frac{f}{p_{b_{i,j}}'}. \tag{108}$$

The continuity equation is solved first. The pressure gradient of the equation of motion (106) uses the value of the state of perturbation η coming from this calculation. The combination of the forward time-stepping in the continuity equation and backward time-stepping in the momentum equations is called the forward-backward scheme.

6.2 Usage

In Micom 2.6, the numerical calculation of the evolution of the barotropic mode is performed in the subprogram :

`subroutine barotp` (argument list)

6.2.1 Order of operations

The new values of the layer thicknesses coming from the continuity equation are introduced by calculating the pressures at interfaces :

```
c
      do 12 j=1,jj1
      do 12 k=1,kk
      do 12 l=1,isp(j)
      do 12 i=ifp(j,l),ilp(j,l)
   12 p(i,j,k+1)=p(i,j,k)+dp(i,j,k+nn)
c
```

Then, for each velocity component, the next step is to rescale, at each point, the thicknesses of N layers by applying the procedure formulated in (97) and to effect the rearrangement of the vertical velocity profile :

```
c
c --- compute new -dpu,dpv- field.  save old -dpu,dpv- values in -pu,pv-.
c --- substitute depth-weighted averages for (u,v) at massless grid points.
c --- (scan layers in top-down direction to save time.)
c
      do 14 j=1,jj1
c
      do 13 l=1,isu(j)
      do 13 k=1,kk
      kn=k+nn
c
      do 113 i=ifu(j,l),ilu(j,l)
      pu(i,j,k+1)=dpu(i,j,kn)
  113 dpu(i,j,kn)=max(0.,
     .  min(depthu(i,j),.5*(p(i,j,k+1)+p(i-1,j,k+1)))-
     .  min(depthu(i,j),.5*(p(i,j,k )+p(i-1,j,k ))))
c
      if (k.gt.1) then
      do 123 i=ifu(j,l),ilu(j,l)
      q=tencm-min(dpu(i,j,kn),tencm)
  123 u(i,j,kn)=(u(i,j,kn)*dpu(i,j,kn)+u(i,j,kn-1)*q)/(dpu(i,j,kn)+q)
      end if
c
   13 continue
c
      do 14 l=1,isv(j)
      do 14 k=1,kk
      kn=k+nn
c
      do 114 i=ifv(j,l),ilv(j,l)
```

```
      pv(i,j,k+1)=dpv(i,j,kn)
  114 dpv(i,j,kn)=max(0.,
     .  min(depthv(i,j),.5*(p(i,j,k+1)+p(i,j-1,k+1)))-
     .  min(depthv(i,j),.5*(p(i,j,k )+p(i,j-1,k ))))
c
      if (k.gt.1) then
      do 124 i=ifv(j,l),ilv(j,l)
      q=tencm-min(dpv(i,j,kn),tencm)
  124 v(i,j,kn)=(v(i,j,kn)*dpv(i,j,kn)+v(i,j,kn-1)*q)/(dpv(i,j,kn)+q)
      end if
c
   14 continue
c
```

From these profiles we extract the corresponding components of the pseudo-velocity $\overline{\mathbf{u}}^*$ by the calculation of sums given by (104) :

```
c
c --- extract barotropic velocities generated during most recent baroclinic
c --- time step and use them to force barotropic flow field.
c
      do 30 j=1,jj1
c
      do 31 l=1,isu(j)
c
      do 32 i=ifu(j,l),ilu(j,l)
   32 utotn(i,j)=0.
      do 33 k=1,kk
      kn=k+nn
      do 33 i=ifu(j,l),ilu(j,l)
   33 utotn(i,j)=utotn(i,j)+u(i,j,kn)*dpu(i,j,kn)
      do 31 i=ifu(j,l),ilu(j,l)
   31 utotn(i,j)=utotn(i,j)/depthu(i,j)
c
      do 30 l=1,isv(j)
c
      do 34 i=ifv(j,l),ilv(j,l)
   34 vtotn(i,j)=0.
      do 35 k=1,kk
      kn=k+nn
      do 35 i=ifv(j,l),ilv(j,l)
   35 vtotn(i,j)=vtotn(i,j)+v(i,j,kn)*dpv(i,j,kn)
      do 30 i=ifv(j,l),ilv(j,l)
   30 vtotn(i,j)=vtotn(i,j)/depthv(i,j)
c
```

The final operation consists of using the Asselin filtering on the baroclinic velocity components $\mathbf{u}'$:

```
c
c --- time smoothing of -u,v- fields
c
      do 22 k=1,kk
      km=k+mm
      kn=k+nn
c
      do 22 j=1,jj1
c
      do 24 l=1,isu(j)
      do 24 i=ifu(j,l),ilu(j,l)
      u(i,j,kn)=u(i,j,kn)-utotn(i,j)
   24 u(i,j,km)=(u(i,j,km)*(wuv1*dpu(i,j,km)+onemm)
     .  +wuv2*(uold(i,j,k)*pu(i,j,k+1)+u(i,j,kn)*dpu(i,j,kn)))/
     .  (wuv1*dpu(i,j,km)+onemm+wuv2*(pu(i,j,k+1)+dpu(i,j,kn)))
c
      do 22 l=1,isv(j)
      do 22 i=ifv(j,l),ilv(j,l)
      v(i,j,kn)=v(i,j,kn)-vtotn(i,j)
   22 v(i,j,km)=(v(i,j,km)*(wuv1*dpv(i,j,km)+onemm)
     .  +wuv2*(vold(i,j,k)*pv(i,j,k+1)+v(i,j,kn)*dpv(i,j,kn)))/
     .  (wuv1*dpv(i,j,km)+onemm+wuv2*(pv(i,j,k+1)+dpv(i,j,kn)))
c
```

In the step which follows, we perform the treatment of the equations of continuity and motion $\mathcal{N}$ times, as we described previously (*cf.* § 6.1.4 and § 6.1.5). The new values of the barotropic component $\overline{\mathbf{u}}$ and of the state of perturbation η are calculated using the Asselin weighting factor w :

```
c
c --- ------------------------------------------------------------------
c --- advance barotropic equations from baroclinic time level -m- to level -n-
c --- ------------------------------------------------------------------
c
      do 867 j=1,jj1
c
      do 865 l=1,isu(j)
      do 865 i=ifu(j,l),ilu(j,l)
      utotn(i,j)=utotn(i,j)/delt1
  865 ubavg(i,j,n)=ubavg(i,j,m)
c
      do 866 l=1,isv(j)
      do 866 i=ifv(j,l),ilv(j,l)
      vtotn(i,j)=vtotn(i,j)/delt1
```

```
  866 vbavg(i,j,n)=vbavg(i,j,m)
c
      do 867 l=1,isp(j)
      do 867 i=ifp(j,l),ilp(j,l)
  867 pbavg(i,j,n)=pbavg(i,j,m)
c
      ml=n
      nl=3
c
c --- explicit time integration of barotropic flow (forward-backward scheme)
c --- in order to combine forward-backward scheme with leapfrog treatment of
c --- coriolis term, v-eqn must be solved before u-eqn every other time step
c
      do 840 lll=1,lstep
c
c --- continuity equation
c
      do 843 j=1,jj1
      do 843 l=1,isp(j)
      do 843 i=ifp(j,l),ilp(j,l)
c --- use light time smoother in continuity equation
  843 pbavg(i,j,nl)=(1.-wbaro)*pbavg(i,j,ml)+wbaro*pbavg(i,j,nl)
     .  -(1.+wbaro)*dlt*(ubavg(i+1,j,ml)*depthu(i+1,j)*scu(i+1)
     .  -ubavg(i ,j,ml)*depthu(i ,j)*scu(i )
     .  +vbavg(i,j+1,ml)*depthv(i,j+1)*scv(i )
     .  -vbavg(i,j ,ml)*depthv(i,j )*scv(i ))*scp2i(i)
c
      mn=ml
c
c --- u momentum equation
c
      do 841 j=1,jj1
      do 841 l=1,isu(j)
      do 841 i=ifu(j,l),ilu(j,l)
      utndcy=-thref*(pbavg(i,j,nl)-pbavg(i-1,j,nl))*scui(i)
     .+(vbavg(i ,j,mn)*depthv(i ,j)+vbavg(i ,j+1,mn)*depthv(i ,j+1)
     .  +vbavg(i-1,j,mn)*depthv(i-1,j)+vbavg(i-1,j+1,mn)*depthv(i-1,j+1))
     .  *(pvtrop(i,j)+pvtrop(i,j+1))*.125
  841 ubavg(i,j,nl)=(1.-wbaro)*ubavg(i,j,ml)+wbaro*ubavg(i,j,nl)
     .  +(1.+wbaro)*dlt*(utndcy+utotn(i,j))
c
      mn=nl
c
c --- v momentum equation
c
      do 842 i=1,ii1
      do 842 l=1,jsv(i)
```

```
      do 842 j=jfv(i,l),jlv(i,l)
      vtndcy=-thref*(pbavg(i,j,nl)-pbavg(i,j-1,nl))*scvi(i)
     .-(ubavg(i,j ,mn)*depthu(i,j )+ubavg(i+1,j ,mn)*depthu(i+1,j )
     .  +ubavg(i,j-1,mn)*depthu(i,j-1)+ubavg(i+1,j-1,mn)*depthu(i+1,j-1))
     .  *(pvtrop(i,j)+pvtrop(i+1,j))*.125
  842 vbavg(i,j,nl)=(1.-wbaro)*vbavg(i,j,ml)+wbaro*vbavg(i,j,nl)
     .  +(1.+wbaro)*dlt*(vtndcy+vtotn(i,j))
c
      mn=nl
c
      ll=ml
      ml=nl
      nl=ll
c
  840 continue
c
```

6.2.2 Flowchart

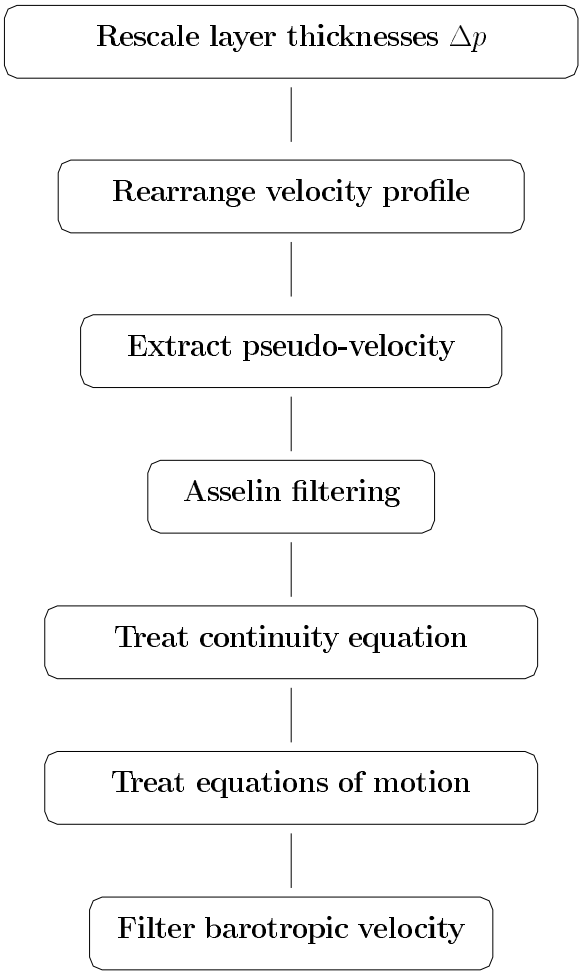

Figure 11: *Order of the barotropic mode calculation*

6.3 Variables

6.3.1 Identification

Notation in the theory	Notation in **barotp.f**
$\Delta t_B, \Delta t_b$	`dlt,delt1`
$f'_{i,j}$	`pvtrop(i,j)`
$\gamma, (1-2\gamma)$	`wuv2,wuv1`
$\mathcal{N}$	`lstep`
$p''_{b_{i,j}}$	`pbavg(i,j)`
${}_u p'_{b_{i,j}}, {}_u p'_{b_{i,j}}$	`depthu(i,j),depthv(i,j)`
${}_u \Delta p'_{i,j,k}, {}_u p'_{i,j,k}$	`dpu(i,j,k),dpv(i,j,k)`
S_u, S_v	`utotn(i,j),vtotn(i,j)`
$\overline{u}^n_{i,j}, \overline{v}^n_{i,j}$	`ubavg(i,j,n),ubavg(i,j,n)`
w	`wbaro`

6.3.2 Global variables

integer variables : `ifp(j,m)` `ilp(j,m)`, `ifv(j,m)`, `ilv(j,m)`, `isp(j)`, `isu(j)`, `isv(j)`, `jj1(j)`, `kk`, `lstep`

real variables : `depthu(i,j)`, `depthv(i,j)`, `dpu(i,j,k)`, `dpv(i,j,k)`, `dp(i,j,k)`, `onemm`, `p(i,j,k)`, `pbavg(i,j,n)`, `pu(i,j,k)`, `pv(i,j,k)`, `pvtrop(i,j)`, `scu(i)`, `scv(i)`, `tencm`, `u(i,j,k)`, `ubavg(i,j,n)`, `uold(i,j,k)`, `utotn(i,j,k)`, `v(i,j,k)`, `vbavg(i,j,n)`, `vold(i,j,k)`, `vtotn(i,j,k)`, `wbaro`, `wuv1`, `wuv2`,

logical variables : `vthenu`

6.3.3 Local variables

`i,j,k,l,lll`	loop counters
`km,kn,ml,nl`	intermediate indices
`utndcy,vtndcy`	tendency of the barotropic velocity components

7 Convection : convec.f

In MICOM 2.6, the variations of mixed layer characteristics result from 2 principal steps :

1) explicitly taking into account ocean-atmosphere exchanges (*c.f.* § 9) ;

2) modeling advection with the velocity field (*c.f.* § 9).

Consequently, an inversion at the base of the mixed layer is not excluded. Phase 1) is optional. When radiative exchanges and heat turbulence are not considered, the mechanical effect due to the wind can always be treated through the surface Reynolds tension (*c.f.* § 4). The advection step is systematically used. It can, by itself, generate a surface inversion.

7.1 Usage

In MICOM 2.6, within subroutine `convec`, for each point, we diagnose this possible inversion, determine its vertical extension and address it by conserving the content of heat and salt in the layers concerned.

7.1.1 Order of operations

The first step consists of traversing the vertical and testing the value of the density deviation $\rho_k - \rho_1$. When it is negative, the situation is unstable and we address it by mixing layer k with the layer above it, conserving heat and salt. Then we proceed to infer the density ρ'_1 which serves as a new reference to test for a possible inversion with layer $k+1$.

```
c
      do 3 k=2,kk
      kn=k+nn
      dthmin=999.
c
      do 4 l=1,jsp(i)
      do 4 j=jfp(i,l),jlp(i,l)
      dthet=theta(k)-thmix(i,j,n)
      if (dthet.lt.0..and.dp(i,j,kn).gt.0.)  then
c
c --- layer -k- contains mass less dense than mixed layer.  entrain.
      delp=dp(i,j,k1n)+dp(i,j,kn)
      saln(i,j,k1n)=(saln(i,j,k1n)*dp(i,j,k1n)
      .  +saln(i,j,kn )*dp(i,j,kn ))/delp
      temp(i,j,k1n)=(temp(i,j,k1n)*dp(i,j,k1n)
      .  +temp(i,j,kn )*dp(i,j,kn ))/delp
      thmix(i,j,n)=sig(temp(i,j,k1n),saln(i,j,k1n))-thbase
c
c --- mass in layer -k- transferred to mixed layer is stored in -dpold-
      dp(i,j,k1n)=delp
      dpold(i,j,k)=dp(i,j,kn)
      dp(i,j,kn)=0.
      end if !  dthet < 0
```

```
c
    4 dthmin=min(dthmin,dthet)
c
      if (dthmin.ge.0.)  go to 1
    3 kk1(i)=k
c
```

The second step consists of conserving the momentum by integrating it over the new surface layer.

```
c
c --- entrain -u- momentum
c
      do 7 i=1,ii1
c
      do 5 l=1,jsu(i)
c
      do 6 j=jfu(i,l),jlu(i,l)
    6 util2(i,j)=min(.5*(dpold(i,j,1)+dpold(i-1,j,1)),depthu(i,j))
c
      do 5 k=2,max(kk1(i),kk1(i-1))
      kn=k+nn
      do 5 j=jfu(i,l),jlu(i,l)
      util1(i,j)=util2(i,j)
      util2(i,j)=min(util2(i,j)+
     .  .5*(dpold(i,j,k)+dpold(i-1,j,k)),depthu(i,j))
    5 u(i,j,k1n)=(u(i,j,k1n)*util1(i,j)
     .  +u(i,j,kn)*(util2(i,j)-util1(i,j)))/util2(i,j)
c
c --- entrain -v- momentum
c
      do 7 l=1,jsv(i)
c
      do 8 j=jfv(i,l),jlv(i,l)
    8 util2(i,j)=min(.5*(dpold(i,j,1)+dpold(i,j-1,1)),depthv(i,j))
c
      do 7 k=2,kk1(i)
      kn=k+nn
      do 7 j=jfv(i,l),jlv(i,l)
      util1(i,j)=util2(i,j)
      util2(i,j)=min(util2(i,j)+
     .  .5*(dpold(i,j,k)+dpold(i,j-1,k)),depthv(i,j))
    7 v(i,j,k1n)=(v(i,j,k1n)*util1(i,j)
     .  +v(i,j,kn)*(util2(i,j)-util1(i,j)))/util2(i,j)
c
```

The final operation consists of storing this new vertical distribution.

```
c
      do 66 j=1,jj1
      do 66 k=1,kk
      do 66 l=1,isp(j)
      do 66 i=ifp(j,l),ilp(j,l)
   66 p(i,j,k+1)=p(i,j,k)+dp(i,j,k+nn)
c
```

7.2 Variables

7.2.1 Global variables

integer variables : `ii1`, `jfp(i,l)`, `jlp(i,l)`, `jsp(i)`, `jfu(i,l)`, `jlu(i,l)`, `jsu(i)`, `jfv(i,l)`, `jlv(i,l)`, `jsv(i)`, `kk`

real variables : `depthu(i,j)`, `depthv(i,j)`, `dp(i,j,k)`, `dpold(i,j,k)`, `p(i,j,k)`, `saln(i,j,n)`, `theta(k)`, `thmix(i,j,n)`, `temp(i,j,n)`, `u(i,j,k)`, `util1(i,j)`, `util2(i,j)`, `v(i,j,k)`

7.2.2 Local variables

`ia,ib,ja,jb`	intermediate indices
`delp`	intermediate value of the mixed layer thickness
`dthet`	density deviation between the tested layer and the surface layer
`dthmin`	minimum value of the density deviation permitting the iterative calculation over the vertical
`kk1(idm)`	value of the last layer concerned with the convection mechanism in the computational row

8 Diapycnal mixing : diapfl.f

8.1 Formalism and numerical techniques

8.1.1 Turbulent diffusion

Using θ to represent the vertical distribution of density in the ocean, we use the standard concept of turbulent diffusion to express the turbulent fluxes associated with this variable of state :

$$\overline{w'\theta'} = K_H \partial\theta/\partial z \tag{109}$$

where K_H is the diffusivity. w' and θ' represent the turbulent fluctuations in the vertical component of velocity and in density in the Reynolds sense. With neither a source term, diffusion, nor horizontal advection, the evolution equation of the variable θ at depth z is then written as :

$$\left(\frac{\partial\theta}{\partial t}\right)_z = \frac{\partial}{\partial z}\left(K_H \frac{\partial\theta}{\partial z}\right) \tag{110}$$

The local tendency of the variable θ then results simply from the divergence of vertical turbulent flux associated with this parameter.
Set $F = K_H \partial\theta/\partial z$. Under the assumption that the turbulent flux is zero at the surface ($F_s = 0$) and at the bottom ($F_b = 0$), integrating equation (110) over the vertical conserves the quantity θ in a column of water.
In the hypothetical case where the flux at the boundary is zero, since we don't take into account a source term such as solar radiation, the vertical component of the turbulent flux F can result only from the initial profile of the parameter θ. Over the profile at time t, this flux will cause a vertical displacement of the isocline θ, situated at the the level z which is described by the equation :

$$\left(\frac{\partial z}{\partial t}\right)_\theta = -\frac{\partial F}{\partial\theta} \tag{111}$$

The variable θ is assumed to be stepwise constant in the vertical :

$$\theta(z) = \theta_k \quad \text{for} \quad z_{k-1/2} \leq z \leq z_{k+1/2}$$

The indices $k-1/2$ and $k+1/2$ are the positions of the interfaces above and below the layer k.
In a layer configuration, we want to express the form (111) in centered finite differences considering that the vertical distribution of turbulent flux can also be represented by a succession of layers of flux F_k centered at z_k. We then have :

$$\left(\frac{\partial F}{\partial\theta}\right)_{k+1/2} = -\frac{(F_{k+1} - F_k)}{\theta_{k+1} - \theta_k} \tag{112}$$

The layer scheme therefore imposes the knowledge of the divergence of flux at successive interfaces. To calculate a vertical distribution of fluxes centered at z_k we create an artificial gradient on the interior of each layer k.
The diapycnic mixing algorithm implemented in MICOM is a simplified version of the algorithm developed by Hu (1991).

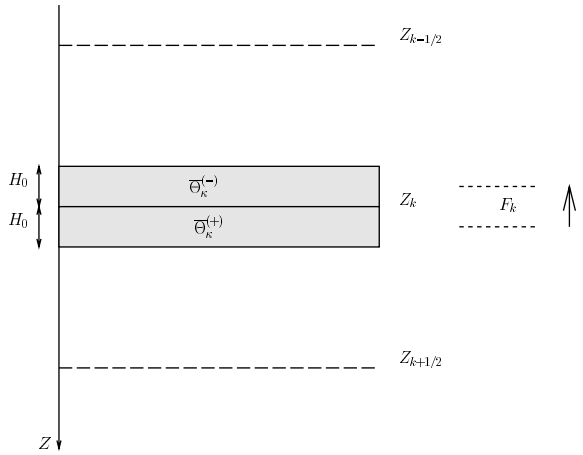

Figure 12: *Illustration of the calculation of turbulent flux F_k.*

8.1.2 Turbulent heat flux

To illustrate the mechanism of diapycnic mixing, we can imagine that the turbulent instability due to the temperature discontinuity at the interface between two layers manifests itself in the appearance of a thin intermediate mixed layer. At the upper limit of the layer k, at the coordinate $z_{k-1/2}$, the temperature of this micro-layer is posited to be the same as θ'_{sup} such that :

$$\theta'_{sup} = \frac{1}{2}\left(\theta_k + \theta_{k-1}\right) \tag{113}$$

Then, we introduce the continuous function by breaking up :

$$\theta'(z) = \frac{1}{z_{k+1/2} - z_{k-1/2}} \left[\frac{\theta_k + \theta_{k-1}}{2}(z_{k+1/2} - z) + \frac{\theta_k + \theta_{k+1}}{2}(z - z_{k-1/2}) \right] \tag{114}$$

for : $z_{k-1/2} \leq z \leq z_{k+1/2}$. The distribution $\theta'(z)$ linearly varies the influence of the boundary conditions of the diapycnic mixing on the interior of layer k in such a manner that the influence of the upper boundary condition is zero at the lower interface and *vice versa*. At the surface, we represent the surface mixed layer by introducing the upper discontinuity : $\theta'(z) = \theta_1$ for $z \leq z_{3/2}$.

We integrate the new distribution $\theta'(z)$ over two intervals of finite thickness H_0, situated respectively above and below the middle point of layer k of depth $z_k = 1/2\left(z_{k-1/2} + z_{k+1/2}\right)$. We obtain the two new variables (figure 12):

$$\overline{\theta}_k^{(-)} = \frac{1}{H_0} \int_{z_k - H_0}^{z_k} \theta' \, dz \tag{115}$$

$$\overline{\theta}_k^{(+)} = \frac{1}{H_0} \int_{z_k}^{z_k + H_0} \theta' \, dz \tag{116}$$

We then define the flux in k as

$$F_k = K_H \frac{\overline{\theta}_k^{(+)} - \overline{\theta}_k^{(-)}}{H_0} \tag{117}$$

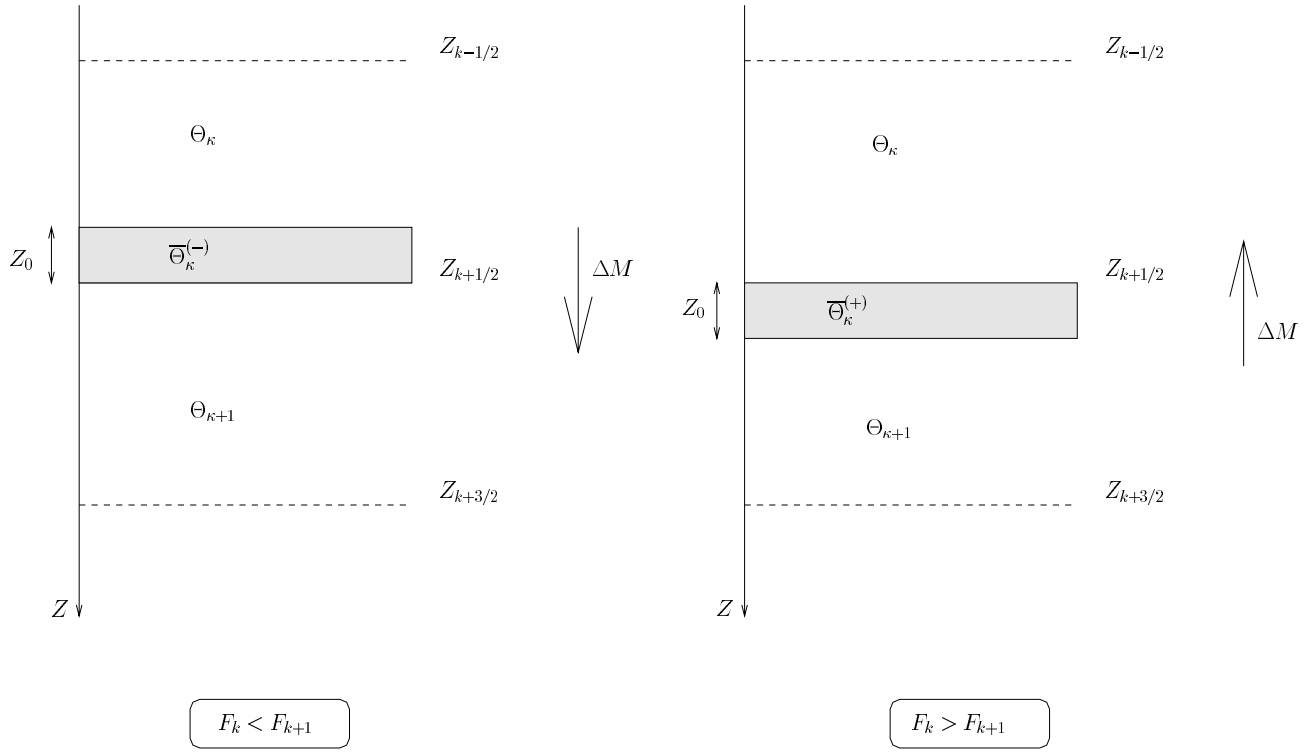

Figure 13: *Illustration of the calculation of the divergence of the turbulent flux F_k.*

Note that for layers of thickness greater than $2H_0$, we have the relation :

$$F_k = K_H \frac{\theta_{k+1} - \theta_{k-1}}{2(z_{k+1/2} - z_{k-1/2})}$$

8.1.3 Numerical implementation

During a time interval Δt, after having been identified with the relation (111), the form (112) expresses that a column of water of thickness $\Delta z = (\partial z/\partial t)\,\Delta t$ is going to see its density change from θ_{k+1} to θ_k (or *vice-versa* depending on the sign of $\partial F/\partial\theta$). In other words, in the deep regime, during Δt, the diapycnic mixing causes a vertical transfer of the quantity Δz of the layer k to the layer $k+1$. Numerically, this transfer can pose problems if Δz exceeds the thickness of layer k. This fact most concerns layers of small thickness. It can also appear during a transfer of water from the mixed layer to the first layer below where the increment $(\theta_k - \theta_{k+1})$ is often small.

Consider the case $F_k > F_{k+1}$. We want to do a mass transfer from layer $k+1$ to layer k. To generalize the expression 112) HU (1991) substitutes θ_{k+1} with the average expression (figure 13) :

$$\overline{\theta}^{(+)}_{k+1/2} = Z_0^{-1} \int_{z_{k+1/2}}^{z_{k+1/2}+Z_0} \theta\, dz \tag{118}$$

where we see the discrete distribution $\theta(z)$ reappear.

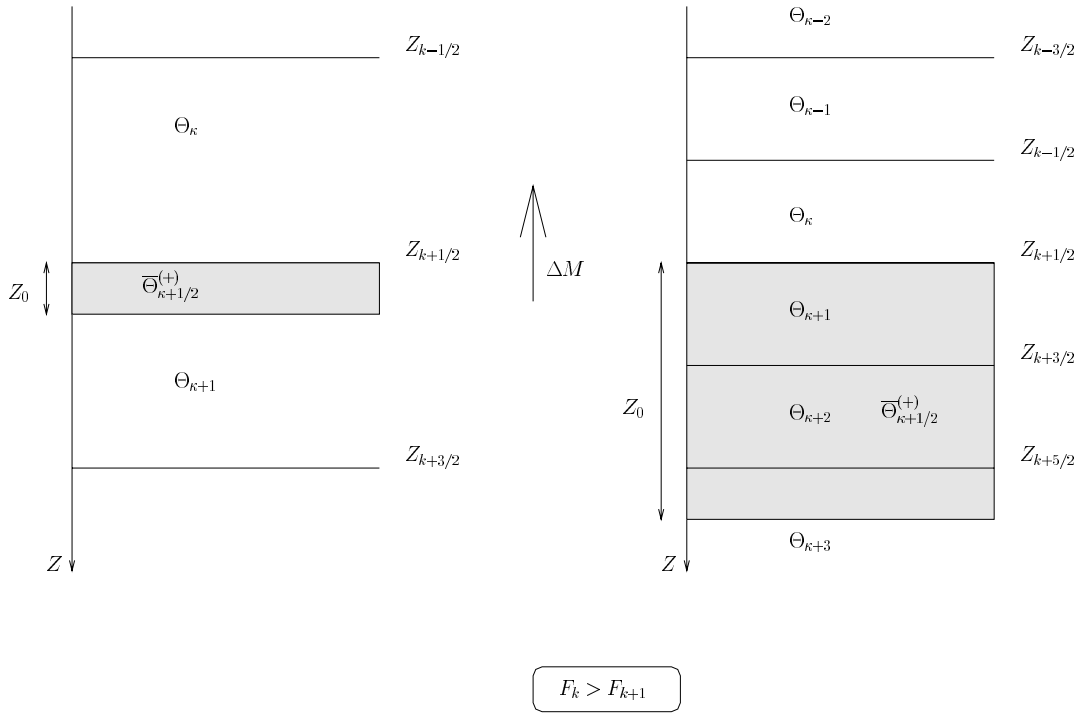

Figure 14: *Illustration of the generalization of* HU *(1991) in the case of an ascending mass transfer.*

In this case, the general form (112) becomes :

$$\left(\frac{\partial F}{\partial \theta}\right)_{k+1/2} = -\frac{(F_{k+1} - F_k)}{\overline{\theta}^{(+)}_{k+1/2} - \theta_k} \tag{119}$$

In identifying this last expression with the relation (111), it is clear that the displacement Δz of the interface $k+1/2$ represents the thickness of a layer whose density evolves from $\overline{\theta}^{(+)}_{k+1/2}$ to θ_k. Only two alternatives are possible (figure 14) :

1) if layer k+1 is thicker than Z_0, then $\overline{\theta}^{(+)}_{k+1/2} = \theta_{k+1}$. Equation (119) takes the form (112). More frequently, to carry out a transfer into the layer k, it is possible to extract the ascending mass only from layer $k+1$;

2) if layer $k+1$ is thinner than Z_0, the approach of HU says that the layers $k+1$, $k+2$,... contribute to the total displacement Δz in the same proportions in which they contribute to the integral (118).

Concerning the case where $F_k < F_{k+1}$, the form (112) is then substituted with :

$$\left(\frac{\partial F}{\partial \theta}\right)_{k+1/2} = -\frac{(F_{k+1} - F_k)}{\theta_{k+1} - \overline{\theta}^{(-)}_{k+1/2}} \tag{120}$$

where,

$$\overline{\theta}^{(-)}_{k+1/2} = Z_0^{-1} \int_{z_{k+1/2}-Z_0}^{z_{k+1/2}} \theta \, dz. \tag{121}$$

The mass of the column of water of thickness Δz and of density $\overline{\theta}^{(-)}_{k+1/2}$ transferred from layer k into layer $k+1$ comes from layers k, k-1, k-2,... in the same proportions in which they contribute to the integral (121).

8.2 Usage

In MICOM, the numerical calculation of diapycnic mixing is carried out by the subprogram :

`subroutine diapfl` (argument list)

8.2.1 Order of operations

For each point of the calculation, we determine the index of the first isopycnic layer of which the specific volume is strictly larger than that of the mixed layer. The result is stored in the array `klist(i,j)`. Then, we evaluate an adequate integration thickness H_0. The result is stored in the array `dpmx(i,j)`. In the first step, the operation consists of finding two limiting values : `h0lo` in the surface and `h0hi` from a fixed pressure `delp`. These values themselves are bounded by 1/4 of the total water height. Between these two depths, the final value varies linearly.

```
c
      do 17 i=ifp(j,l),ilp(j,l)
      sgain(i,k)=saln(i,j,kn)*dp(i,j,kn)
c --- store index of first isopycnic layer heavier than mix.layer in -klist-
      if (theta(k).lt.thmix(i,j,n)) klist(i,j)=min(k+1,kk)
c
c --- dpmx increases from -h0lo- to -h0hi- over depth range -delp-
      delp=1000.*onem
      h0lo=min( h0,.25*p(i,j,kk+1))
      h0hi=min(5.*h0,.25*p(i,j,kk+1))
      wgt=max(0.,min(1.,(delp-p(i,j,k-1))/delp))
      dpmx(i,j)=h0lo*wgt+h0hi*(1.-wgt)
c
      thup(i,k)=epsil*max(thmix(i,j,mn),theta(k-1))
      thdn(i,k)=epsil*max(thmix(i,j,mn),theta(k ))
      thup(i,1)=thdn(i,k)
      thdn(i,1)=thdn(i,k)
      util1(i,j)=epsil
      util2(i,j)=epsil
      util3(i,j)=epsil
   17 util4(i,j)=epsil
c
```

Then, we explore the part above in the column of water to find layers which will comprise the interval Z_0 and we calculate $\overline{\theta}^{(-)}_{k+1/2}$ over this slice of water. Next, we explore the part below in the column of water to find the layers which will comprise the interval Z_0 and we calculate $\overline{\theta}^{(+)}_{k+1/2}$ over this slice of water. The results are stored in the arrays `thup(i,k)` and `thdn(i,k)`. The thickness Z_0 is also put to the initial value H_0. Similarly if it is small (10 meters for ocean applications), it is possible that the instantaneous thickness of a layer be even smaller, or zero.

The algorithm accounts for this.

```
c
c --- find bulk theta value in slab of thickness -h0- above k-th interface
      do 61 kp=1,k-1
      do 61 i=ifp(j,l),ilp(j,l)
      delp=max(0.,min(p(i,j,k) ,p(i,j,kp+1))
     .  -max(p(i,j,k)-h0,p(i,j,kp )))
      thup(i,k)=thup(i,k)+max(thmix(i,j,mn),theta(kp))*delp
   61 util1(i,j)=util1(i,j) +delp
c
c --- find bulk theta value in slab of thickness -h0- below k-th interface
      do 51 kp=k,kk
      do 51 i=ifp(j,l),ilp(j,l)
      delp=max(0.,min(p(i,j,k)+h0,p(i,j,kp+1))
     .  -max(p(i,j,k) ,p(i,j,kp )))
      thdn(i,k)=thdn(i,k)+max(thmix(i,j,mn),theta(kp))*delp
   51 util2(i,j)=util2(i,j)
c
```

The next operation consists of determining the two temperatures $\overline{\theta}_k^{(-)}$ and $\overline{\theta}_k^{(+)}$ defined by the expressions (115) and (116). For this, we inspect the whole column of water to calculate a bulk characteristic temperature of each of two slices of water situated respectively above and below an interior point in layer k. The thickness `dpmx(i,j)` of each of the two intermediate layers is limited by half of the vertical extension of the layer studied. It is also necessary to account for the fact that a layer can have zero size.

```
c
      do 36 i=ifp(j,l),ilp(j,l)
c
      thk= max(thmix(i,j,mn),theta(kp))
      thka=max(thmix(i,j,mn),theta(ka))
      thkb=max(thmix(i,j,mn),theta(kb))
      if (kp.eq.1) thkb= thk
      if (kp.eq.kk) thkb=2.*thk-thka
      thka=.5*(thk+thka)
      thkb=.5*(thk+thkb)
c
c --- find bulk theta value in slab of thickness -dpmx- above k-th mass point
      pmid=.5*(p(i,j,k)+p(i,j,k+1))
      p1=max(pmid-dpmx(i,j),p(i,j,kp ))
      p2=min(pmid ,p(i,j,kp+1))
      delp=max(0.,p2-p1)
      thup(i,1)=thup(i,1)+(thka*(p(i,j,kp+1)-.5*(p1+p2))
     .  -thkb*(p(i,j,kp )-.5*(p1+p2)))
     .  /max(epsil,p(i,j,kp+1)-p(i,j,kp))*delp
      util3(i,j)=util3(i,j) +delp
```

```
c
c --- find bulk theta value in slab of thickness -dpmx- below k-th mass point
c --- (achieve thdn => thup near sea floor by elevating -pmid-)
      pmid=min(pmid,p(i,j,kk+1)-dpmx(i,j))
      p1=max(pmid ,p(i,j,kp ))
      p2=min(pmid+dpmx(i,j),p(i,j,kp+1))
      delp=max(0.,p2-p1)
      thdn(i,1)=thdn(i,1)+(thka*(p(i,j,kp+1)-.5*(p1+p2))
     .  -thkb*(p(i,j,kp )-.5*(p1+p2)))
     .  /max(epsil,p(i,j,kp+1)-p(i,j,kp))*delp
   36 util4(i,j)=util4(i,j)+delp
c
```

The corresponding flux is calculated by using the relation (117).

```
c
      do 18 i=ifp(j,l),ilp(j,l)
c
      thup(i,k)=thup(i,k)/util1(i,j)
      thdn(i,k)=max(theta(k),thdn(i,k)/util2(i,j))
      thup(i,1)=thup(i,1)/util3(i,j)
      thdn(i,1)=thdn(i,1)/util4(i,j)
c
c --- store turbulent buoyancy fluxes in -dpold- (positive = downward)
      dpold(i,j,k)=diapyc*sqrt(max(0.,thdn(i,1)-thup(i,1))
     .  /(thref*dpmx(i,j)))
   18 if (k.lt.klist(i,j)) dpold(i,j,k)=0.
c
```

In the last step, we first use the generalization of Hu (1991) :

a) if $F_{k-1} - F_k < 0$, we substitute θ_{k-1} by $\overline{\theta}^{(-)}_{k-1/2}$;

b) if $F_{k-1} - F_k \geq 0$, we substitute θ_k by $\overline{\theta}^{(+)}_{k-1/2}$

then, we calculate the corresponding vertical increment Δz. The result is stored in the array `sdot(i,j)`. The maximal positive excursion of the lower interface is bounded by the ocean depth. A negative displacement of the upper interface is limited by 90% of the initial immersion of this interface.

```
c
      do 58 i=ifp(j,l),ilp(j,l)
c
      dif=dpold(i,j,k-1)-dpold(i,j,k)
c --- util1/util2:  upper/lower theta values used in calculating flux div.
      util1(i,j)=max(thmix(i,j,mn),theta(k-1))
      if (dif.lt.0.)  then
      if (dif.lt.0.)  then
```

```
      util1(i,j)=thup(i,k)
      else
      util2(i,j)=thdn(i,k)
      end if
      dthet=max(1.e-5*thref,util2(i,j)-util1(i,j))
c --- sdot = coordinate surface displacement caused by diapyc.  flux.div.
      sdot(i,j)=baclin*mixfrq*onecm*dif/dthet
c
c --- interfaces must not descend below sea floor.  adjust fluxes accordingly
      thkn=p(i,j,k+1)-p(i,j,k)
      if (thkn.ge.0.)  then
      bound=p(i,j,kk+1)-p(i,j,k)
      else
      bound=thkn
      end if
      sdot(i,j)=min(bound,sdot(i,j))
c
c --- interface must not ascend above sea surface
      sdot(i,j)=max(-.9*p(i,j,k),sdot(i,j)) c
      if (k.gt.klist(i,j))
     .dpold(i,j,k-1)=dpold(i,j,k)+sdot(i,j)*dthet/(baclin*mixfrq*onecm)
c
   58 continue
c
```

Finally, we extract the mass of the layers above or below so as to determine the new thickness of each layer concerned.

```
c
c --- sdot < 0 -- extract mass (and salin.)  from layers above k-th interface
      do 59 kp=1,k-1
      kpn=kp+mmnn
      do 59 i=ifp(j,l),ilp(j,l)
      delp=0.
      if (sdot(i,j).lt.0.)
     .delp=max(0.,min(p(i,j,k) ,p(i,j,kp+1))
     .  -max(p(i,j,k)-h0,p(i,j,kp )))*(-sdot(i,j))/h0
c
      dp(i,j,kpn)=dp(i,j,kpn)-delp
      dp(i,j,kn )=dp(i,j,kn )+delp
      saldp=saln(i,j,kpn)*delp
      sgain(i,kp)=sgain(i,kp)-saldp
   59 sgain(i,k )=sgain(i,k )+saldp
c
c --- sdot > 0 -- extract mass (and salin.)  from layers below k-th interface
      do 60 kp=k,kk
      kpn=kp+mmnn
```

```
      do 60 i=ifp(j,l),ilp(j,l)
      delp=0.
      if (sdot(i,j).ge.0.)
     .delp=max(0.,min(p(i,j,k)+h0,p(i,j,kp+1))
     .  -max(p(i,j,k) ,p(i,j,kp )))* sdot(i,j)
     .  /max(epsil,min(p(i,j,kk+1)-p(i,j,k),h0))
c
      dp(i,j,kpn )=dp(i,j,kpn )-delp
      dp(i,j,kn-1)=dp(i,j,kn-1)+delp
      saldp=saln(i,j,kpn)*delp
      sgain(i,kp )=sgain(i,kp )-saldp
   60 sgain(i,k-1)=sgain(i,k-1)+saldp
c
      do 78 k=1,kk
      kn=k+mmnn
c
      do 78 i=ifp(j,l),ilp(j,l)
      p(i,j,k+1)=p(i,j,k)+dp(i,j,kn)
c
c --- get new salinity from salinity integral stored in -sgain-
      if (dp(i,j,kn).gt.onecm)
     .saln(i,j,kn)=sgain(i,k)/dp(i,j,kn)
   78 continue
c
```

To end, we calculate the interface pressures.

```
c
c --- water from ultra-thin layers (typically near bottom) may have repeatedly
c --- been transferred to other layers, causing dp < 0.  adjust.
c
      do 79 k=kk,1,-1
      do 79 i=ifp(j,l),ilp(j,l)
      p(i,j,k)=max(0.,min(p(i,j,k),p(i,j,k+1)))
   79 dp(i,j,k+mmnn)=p(i,j,k+1)-p(i,j,k)
c
```

8.2.2 Flowchart

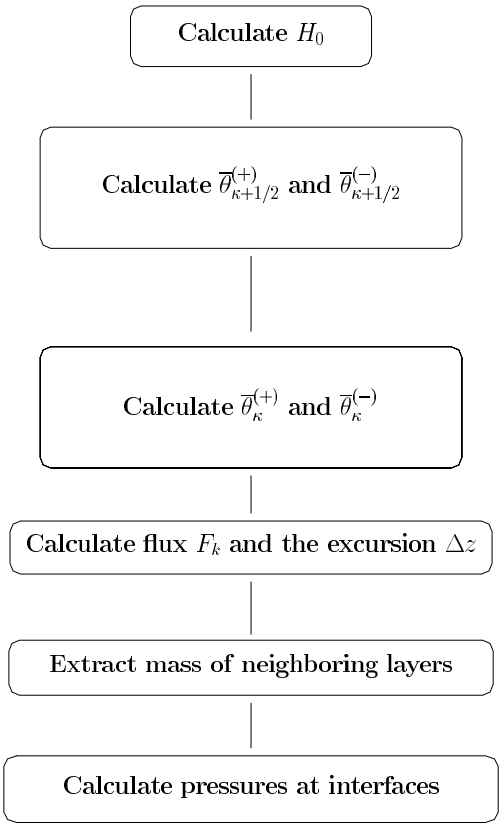

Figure 15: *Order of the mixed layer calculations in MICOM 2.6*

8.3 Variables

8.3.1 Identification

Notation in the theory of HU (1991)	Notation in **diapfl.f**
H_0	`dpmx`
Z_0	`delp`
$\overline{\theta}_k^{(-)}$	`thup(i,1)`
$\overline{\theta}_k^{(+)}$	`thdn(i,1)`
$\overline{\theta}_{k-1/2}^{(-)}$	`thup(i,k)`
$\overline{\theta}_{k+1/2}^{(+)}$	`thdn(i,k)`

8.3.2 Global variables

integer variables : `ifp(j,m)`, `ilp(j,m)`, `isp(j)`, `klist(i,j)`

real variables : `baclin`, `mixfrq`, `diapyc`, `dp(i,j,k)`, `dpmx(i,j)`, `dpold(i,j)`, `epsil`, `h0`, `klist(i,j)`, `p(i,j,k)`, `sdot(i,j)`, `saln(i,j,k)`, `sgain(i,k)`, `thdn(i,k)`, `theta(i,k)`, `thmix(i,j)`, `thup(i,k)`, `util1(i,j)`, `util2(i,j)`, `util3(i,j)`, `util4(i,j)`

logical variables : `thermo`

8.3.3 Local variables

`bound`	upper bound of the lower interface of a layer
`delp`	integration thickness
`dif`	divergence of heat flux
`dthet`	density growth
`h0hi,h0lo`	integration thickness bounds
`ka,kb,kn,kp`	indices of iterative sequences
`p1,p2,pmid`	intermediate pressures
`saldp`	salinity growth
`thk,thka,thkb`	intermediate values of specific volume
`wgt`	weighting coefficient

9 Ocean mixed layer : mxlayr.f

9.1 Formalism

To model the seasonal evolution of the thermal regime of the surface ocean implies including the effects of forcing by the atmosphere, whose fundamental properties are totally different from those of the ocean. The solar influence translates in part to a gain in radiative heat. Moreover, we have known for a long time that the effect of the prevailing winds in the atmospheric boundary layer manifests itself through turbulent transport of momentum in the surface layers. To account explicitly for the processes connected to ocean-atmosphere exchanges in the context of an isopycnic theory, MICOM uses an integral-type model whose development was initiated by KRAUS & TURNER (1967). In such a model of the surface layer behavior, the problem of closing the turbulence equations is greatly simplified since we need simply to determine the turbulent fluxes at the boundaries of the mixed layer. The progressive shrinking of the thickness h of the surface layer by diapycnic mixing is counterbalanced by a mechanism increasing the thickness. For this, we introduce an entrainment velocity w_e at the base of the layer such that :

$$\begin{aligned} w_e &= \frac{dh}{dt} \quad \text{if} \quad \frac{dh}{dt} > 0 \\ w_e &= 0 \quad \text{if} \quad \frac{dh}{dt} \leq 0 \end{aligned} \tag{122}$$

The originality of this type of model is to express the vertical turbulent fluxes at the base of the mixed layer by an equation of the general form :

$$-\left(\overline{a'w'}\right)_{-h} = w_e \Delta a \tag{123}$$

where Δa depicts the discontinuity of the variable at the base of the layer. In this model, the effect of the wind is represented by the surface stress $\boldsymbol{\tau_s}(\tau_{s_x}, \tau_{s_y})$.

9.1.1 Internal energy and turbulent kinetic energy

The evolution equation of the internal energy of a homogeneous layer of temperature θ_s is then given by :

$$h\frac{\partial \theta_s}{\partial t} = \left(\overline{\theta'w'}\right)_{-h} - \left(\overline{\theta'w'}\right)_0 + \frac{1}{\rho C_p}(R_0 - R_h) - \left[K_H\left(\frac{\partial \theta}{\partial z}\right)\right]_{-h} \tag{124}$$

with : $-\left(\overline{\theta'w'}\right)_0 = \frac{P}{\rho C_p}$ and $-\left(\overline{\theta'w'}\right)_{-h} = w_e \Delta\theta$. R_z is the solar radiation at the depth z and P the surface heat losses (infrared radiation, latent and sensible heat flux). $\Delta\theta$ represents the temperature discontinuity at the base of the mixed layer. At the surface, we have, of course, accounted for the flux $\left[K_H\left(\frac{\partial \theta}{\partial z}\right)\right]_0$. If the latter is null, we situate it in the upper limit of the homogeneous layer.

The knowledge of the entrainment velocity necessitates a supplementary equation. In a homogeneous layer model, we carry out this calculation using the integrated equation of turbulent

kinetic energy $E/2$:

$$\underbrace{\frac{1}{2}\int_{-h}^{0}\frac{\partial E}{\partial t}\,dz}_{1} = \underbrace{\left[\overline{\left(\frac{E}{2}+\frac{\pi'}{\rho_0}w'\right)}\right]_{-h}}_{2} - \underbrace{\left[\overline{\left(\frac{E}{2}+\frac{\pi'}{\rho_0}w'\right)}\right]_{0}}_{3} - \underbrace{\int_{-h}^{0}\overline{\mathbf{u}'w'}.(\partial\mathbf{u}/\partial z)\;dz}_{4} + \underbrace{\int_{-h}^{0}\overline{b'w'}\,dz}_{5} - \underbrace{h\epsilon_m}_{6} \tag{125}$$

with :

$\mathbf{u}(u,v)$: horizontal velocity ;
$b = g(\rho_0 - \rho)/\rho_0$: buoyancy ;
π : pressure ;
ϵ_m : TKE average heat dissipated in the mixed layer.

In the complete form, this equation expresses therefore the equilibrium between :

[1] the TKE tendency contained in the mixed layer ;

[2] the TKE flux at the base of the mixed layer ;

[3] The TKE surface flux ;

[4] a production term from cutting the current ;

[5] a consumption term corresponding to the buoyancy of the homogeneous layer ;

[6] a heat dissipation term.

NILLER & KRAUS (1977) have shown that the TKE flux at the base of the mixed layer is generally negligible. On the other hand, the TKE content of this layer is more often considered constant. Further, following KRAUS & TURNER (1967), the surface TKE flux originating principally from the turbulent agitation at the surface (waves, swells, ...) is parametrized as :

$$\left[\overline{\left(\frac{E}{2}+\frac{\pi'}{\rho_0}w'\right)}\right]_0 = m_2 u_*^3 \tag{126}$$

where u_* is the surface drag velocity such that :

$$u_*^2\mathbf{x} = \boldsymbol{\tau}_s/\rho \tag{127}$$

As for the production term, we simply write :

$$\int_{-h}^{0}\overline{\mathbf{u}'w'}.(\partial\mathbf{u}/\partial z)\;dz = m_3 u_*^3 \tag{128}$$

In referring to a reference state with temperature T_0, salinity S_0, and in hydrostatic equilibrium, let :

$$\rho = \rho_0[1 - \alpha_T(T - T_0) + \beta_S(S - S_0)] \tag{129}$$

with the expansion coefficients α_T and β_S defined such that :

$$\alpha_T = -\frac{1}{\rho}\left(\frac{\partial\rho}{\partial T}\right)_S \quad \text{et} \quad \beta_S = \frac{1}{\rho}\left(\frac{\partial\rho}{\partial S}\right)_T \tag{130}$$

We have :

$$\int_{-h}^{0} \overline{b'w'}\, dz = -\frac{1}{2} h \left(w_e \Delta b + B(h) \right) \tag{131}$$

$B(h)$ then represents the sum of surface flux $-\left(\overline{b'w'}\right)_0$ and of the increase of buoyancy due to the absorption of solar radiation :

$$B(h) = -\left(\overline{b'w'}\right)_0 + \frac{\alpha_T g}{\rho C_p} \left(R_0 + R_h - \frac{2}{h} \int_{-h}^{0} R(z)\, dz \right) \tag{132}$$

The TKE conservation equation in the mixed layer can then finally be written as :

$$(m_2 + m_3) u_*^3 - \frac{h}{2} \left(B(h) - w_e \Delta b \right) - h \epsilon_m = 0 \tag{133}$$

9.1.2 Parametrization of turbulent dissipation

To treat this 'handicap' and to address a generalization of the parametrization of the turbulent dissipation ϵ, GASPAR (1988) reintroduced the vertical dissipation scale of Kolmogorov l_ϵ such that :

$$\epsilon = E^{3/2} / l_\epsilon \tag{134}$$

Let, then, over the mixed layer :

$$\epsilon_m = u_e^3 / l \tag{135}$$

u_e is a characteristic velocity scale of the turbulence in the mixed layer and l a dissipation length which can be expressed formally as a function of diverse parameters :

$$l = F(h, L, \lambda, L_\Delta, L_N) \tag{136}$$

with :

$L = u_*^3 / B(h)$: Monin-Obukov length
$\lambda = u_* / f$: Ekman length
$L_\Delta = (\Delta u^2 + \Delta v^2) / \Delta b$: relative scale at the entrainment zone
$L_N = \left(E^{1/2} / N \right)_{-h}$: scale of stratification at the base of the mixed layer
N : Brunt-Vaïsälä frequency

9.1.3 A recent prediction model of the mixed layer

Since the dynamic instability present at the base of the mixed layer is typically of a time scale on the order of the inertial period, GASPAR (1988) notes that this phenomenon can be neglected in the seasonal studies. In a parallel way, we will be able to omit therefore the influence of the TKE dissipation produced by this term : L_Δ does not enter in the determination (136) of l. On the other hand, to introduce L_N will not have an important effect if ΔT is very weak. This is rarely the case in the ocean. Finally, (135) can be expressed as :

$$h \epsilon_m = u_e^3 G(h/L, h/\lambda) \tag{137}$$

G is a function of the stability parameter of Monin-Obukov h/L and of the rotation parameter h/λ. Realizing that setting $u_e = u_*$ causes us to underestimate the turbulent velocity scale causing convective deepening, GASPAR (1988) writes first of all :

$$u_e^2 = E_m = \frac{1}{h} \int_0^h E\, dz \tag{138}$$

From the earlier studies of the subject, it appears that the stability parameter can be expressed by the relation :

$$G(h/L, h/\lambda) = \frac{h}{l} = a_1 + a_2 \ \max\left[1, h/(0.4\lambda)\right] \exp\left(h/L\right) \tag{139}$$

Finally, the prediction of h is carried out using the expression (133), in which the turbulent dissipation comes from the forms (137), (138) and (139) :

$$(m_2 + m_3)u_*^3 - \frac{h}{2}\left(B(h) - w_e \Delta b\right) - (h/l)E_m^{3/2} = 0 \tag{140}$$

Moreover, we introduce the turbulent vertical characteristic velocity scale such that :

$$u_w^2 = \frac{1}{h}\int_0^h \overline{w'^2}\, dz = W_m \tag{141}$$

Then, we write the TKE equation at the base of the mixed layer :

$$-\left(\overline{b'w'}\right)_{-h} = -\frac{\partial}{\partial z}\left[\overline{\left(\frac{E}{2} + \frac{p'}{\rho_0}w'\right)}\right]_{-h} - \left[\overline{\mathbf{u}'w'}.\left(\partial \mathbf{u}/\partial z\right)\right]_{-h} - \epsilon_{-h} \tag{142}$$

In consideration of the relative values of each term of this relation, GASPAR puts it in the general entrainment form :

$$h\Delta b w_e = m_1 u_e^2 u_w \tag{143}$$

Finally, let :

$$h\Delta b w_e = m_1 E_m W_m^{1/2} \tag{144}$$

The equation giving W_m is obtained by integrating the tendency equation of $\overline{w'^2}$ over the homogeneous layer. GASPAR obtains the form :

$$\left(\frac{1}{2} - \frac{m_5}{3}\right)\left[h\Delta b w_e + hB(h)\right] = \frac{h}{3}\left(\frac{m_4}{l_p} - \frac{1}{l}\right)E_m^{3/2} + \frac{m_3 m_5}{3}u_*^3 - \frac{m_4 h}{l_p}E_m^{1/2}W_m \tag{145}$$

l_p is a characteristic length in the absence of rotation :

$$\frac{h}{l_p} = a_1 + a_2 \exp\left(h/L\right) \tag{146}$$

The equations (140), (144) and (145) constitute the total system of the Oceanic Mixed Layer model described in the article of GASPAR (1988). The entrainment velocity is calculated numerically from the formula :

$$h\Delta b w_e = \frac{\left[(0.5A_p - C_{p1}S_p)^2 + 2C_4(h/l)^2 A_p S_p\right]^{1/2} - (0.5A_p + C_{p1}S_p)}{C_4(h/l)^2 - C_{p1}} \tag{147}$$

With :

$$A_p = C_{p3}u_*^3 - C_{p1}hB(h) \tag{148}$$

$$S_p = (m_2 + m_3)u_*^3 - \frac{1}{2}hB(h) \tag{149}$$

$$C_{p1} = \left[2(1 - m_5)(l_p/l) + m_4\right]/6 \tag{150}$$

$$C_{p3} = \left[m_4(m_2 + m_3) - (l_p/l)(m_2 + m_3 - m_5 m_3)\right]/3 \tag{151}$$

$$C_4 = 2m_4 m_1^{-2} \tag{152}$$

9.1.4 Entrainment condition

From equation (140), it is clear that entrainment does not manifest itself if the condition $S_p > 0$ is true. In the oceanic mixed layer, the TKE resulting from the production mechanisms less the energy consumed to homogenize the heat gain due to thermal input (all solar radiation + surface losses) should be positive.
On the other hand, the relation (144) implies :

$$W_m > 0 \tag{153}$$

In eliminating $h\Delta bw_e$ of (140) and (145), we obtain :

$$W_m = \frac{2C_{p1}}{m_4} E_m - \frac{C_2 l_p}{m_4 h} u_*^3 E_m^{-1/2} \tag{154}$$

where W_m is expressed as a function of E_m and C_2 is a positive constant :

$$C_2 = [(3 - 2m_5)(m_2 + m_3) - m_5 m_3]/3 \tag{155}$$

W_m is cancelled by :

$$E_{m0} = u_*^2 \left(\frac{C_2 l_p}{2C_{p1} h} \right)^{2/3} \tag{156}$$

The entrainment condition (153) is therefore equivalent to :

$$E_m > E_{m0} \tag{157}$$

Entrainment does not appear if the TKE exceeds a minimum given by the form (156). After having eliminated $h\Delta bw_e$ of (140) and (144), E_m is given by the zero of the function :

$$F(E_m) = \frac{1}{2} m_1 E_m W_m^{1/2} + \frac{h}{l} E_m^{3/2} - S_p \tag{158}$$

in which W_m depends on E_m by the intermediary of the relation (154). For $E_m > E_{m0}$, F is a strictly increasing function of E_m. The entrainment can not appear if :

$$F(E_{m0}) < 0 \tag{159}$$

Entrainment can not exist therefore if the TKE balance S_p is greater than the minimal dissipation (*i.e.* the dissipation associated with E_{m0}, the minimum value of E_m).
Writing

$$A_p = S_p - \frac{h}{l} E_{m0}^{3/2}, \tag{160}$$

the necessary and sufficient condition for entrainment at the base of the mixed layer ($w_e > 0$) is :

$$A_p > 0. \tag{161}$$

In the case where this condition is not satisfied, the hypothesis is that h automatically adjusts itself to maintain $A_p = 0$. When the heat balance $B(h)$ is not zero, this is equivalent to :

$$h = \frac{C_{p3}}{C_{p1}} L \tag{162}$$

Following the theory of NIILER & KRAUS (1977), the equilibrium of the mixed layer is attained by :

$$h = \frac{2mu_*^3}{(-B_0)} \tag{163}$$

B_0 is the surface buoyancy flux. The authors assume that it varies linearly inside the mixed layer and is zero at the base. The Monin-Obukov length is then expressed as $L = u_*^3/\kappa B_0$. κ is the Von Karman constant ($\kappa \simeq 0.4$). To remain in accord with this definition, we have : $m = 1.25$.

9.1.5 Constants and numerical parameters

Referring to diverse earlier studies, GASPAR retains the following values by the different parameters introduced above : $m_1 = 0.45$; $m_2 = 2.6$; $m_3 = 1.9$; $m_4 = 2.3$; $m_5 = 0.6$; $a_1 = 0.6$; $a_2 = 0.3$

9.2 Numerical techniques

9.2.1 Entrainment algorithm

During a time interval Δt, after having detected a deepening (entrainment) ($w_e > 0$), the growth of the thickness of the mixed layer can be written :

$$\Delta h = \frac{E\Delta t}{b_{mix} - b_{sub}} \tag{164}$$

With :

$E\Delta t$: TKE variation in the mixed layer
b_{mix} : buoyancy of the mixed layer of thickness h
b_{sub} : buoyancy of the adjacent isopycnic layer

The form (164) can not be incorporated directly in a numerical calculation (the denominator can be zero). Moreover, this expression assumes that under the mixed layer exists a layer of buoyancy b_{sub} of finite thickness. As there is no *a priori* reason for these two assumptions to be true, BLECK *et al.* (1989) have developed a particular method for solving (164).
In MICOM, calculating $E\Delta t$ is optional. Option (1) uses the formulation of the TKE evolution equation following the method of KRAUS & TURNER (1967) ($m \neq 0$; $n = 0.15$; $s = 0$). Option (2) employs the relation (147).

Let b_1 and z_1 be the buoyancy and depth of the mixed layer at a point P of the domain. We introduce the variables z_k to represent the lower positions with respect to the sub-adjacent isopycnic layers of buoyancy b_k. The potential energy (PE) of the column of water is expressed as :

$$PE = \int_{z_0}^{z_N} bz\, dz = \frac{1}{2}\sum_{k=1}^{N} b_k(z_k^2 - z_{k-1}^2) \tag{165}$$

with $z_0 = 0$. The same column of water, but mixed has the PE :

$$PE_{mix} = b_{mix}\frac{z_N^2}{2} \tag{166}$$

with :

$$b_{mix} = \frac{1}{z_N} \sum_{k=1}^{N} b_k (z_k - z_{k-1}) \tag{167}$$

Also let :

$$PE_{mix} = \frac{z_N}{2} \sum_{k=1}^{N} b_k (z_k - z_{k-1}) \tag{168}$$

In remarking that the kinetic energy used to mix the heat acquired by radiation in the surface layer is the same as the difference (168)-(165) over the depth h, we obtain :

$$h = \frac{2E\Delta t + G(l-1) - b_l z_{l-1}^2}{F(l-1) - b_l z_{l-1}} \tag{169}$$

with : $F(l) = \sum_{k=1}^{l} b_k (z_k - z_{k-1})$ and $G(l) = \sum_{k=1}^{l} b_k (z_k^2 - z_{k-1}^2)$. Note that the calculation of h by such a formulation implies knowledge of the number l of the concerned layers.

9.2.2 Detrainment algorithm

Consider the situation after a time interval Δt in the case where the term $E\Delta t$ becomes negative. The index $()_1$ always serves to identify the mixed layer. Suppose the buoyancy b_1 lies between the two discrete values b_k and b_{k-1} of the chosen initial distribution. In this case, the Monin-Obukov length satisfies the condition $L < z_1$.
In principle, we could describe the buoyancy transfer by the conservation equation :

$$(b_1' - b_1)L = (b_1 - b_k)(z_1 - L). \tag{170}$$

But then, we must satisfy the following two conditions :

condition A : $b_1' < b_{max}$ where b_{max} is the buoyancy which a fictitious mixed layer of thickness L would acquire during a time interval Δt under the influence of B_0. It is a threshold procedure. Its usage permits avoiding surface reheating of the ocean caused by errors in the ocean-atmosphere flux which serve to force the model. b_{max} is given by the expression :

$$b_{max} = b_1 - B_0 \Delta t \left(\frac{1}{L} - \frac{1}{z_1} \right) \tag{171}$$

If the value of b_1' evaluated by (170) exceeds b_{max}, this threshold is translated into the new mixed layer thickness :

$$z_1' = z_1 \frac{b_1 - b_k}{b_{max} - b_k}. \tag{172}$$

condition B : $b_1' < b_{k-1}$. If this condition B is true, it will be necessary to partition the buoyancy transfer over the layers k and $k-1$. This provision handles well the non-isopycnic behavior of the mixed layer between discrete values b_k and b_{k-1}. For $b_1' > b_{k-1}$, it is possible to distribute a part of the buoyancy over a layer of density ρ_{k-1} and of thickness $z_1' - L$ such that :

$$z_{k-1}' = z_1 \frac{b_1 - b_k}{b_{k-1} - b_k} - L \frac{b_{max} - b_{k-1}}{b_{k-1} - b_k} \tag{173}$$

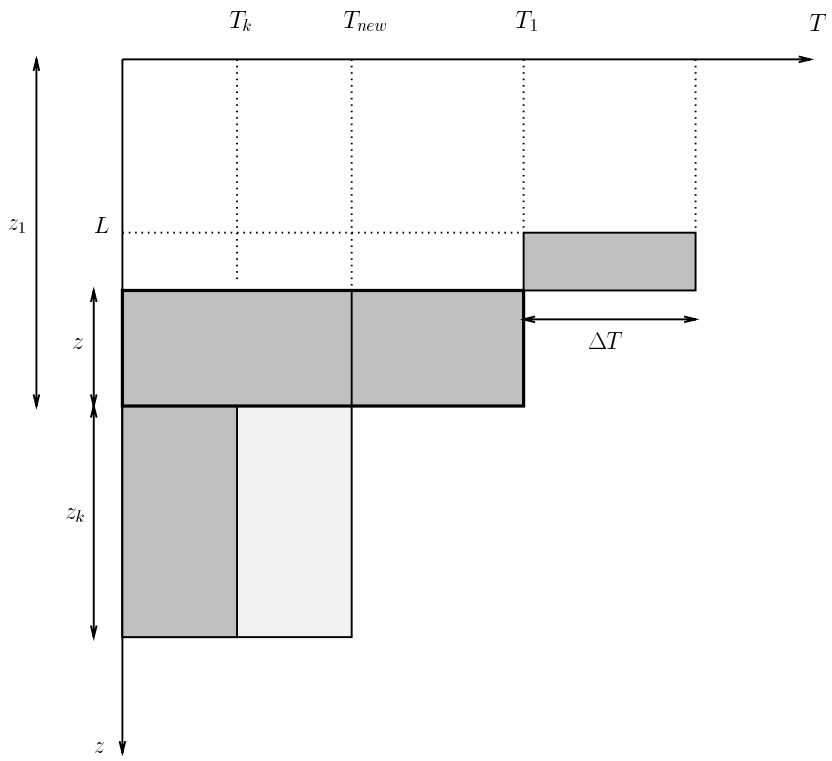

Figure 16: *Conservation of heat during detrainment of the mixed layer.*

If condition B is not satisfied, the mixed-layer detrainment mechanism therefore leads to inflation of two sub-surface layers. We can decompose the process in three steps (figure 16) :

1. formation of the new mixed layer of thickness equal to the Monin-Obukov length and of buoyancy b_{max} ;
2. inflation of an intermediate layer $(k-1)$;
3. adjustment of the old sub-adjacent layer (k) to the mixed layer.

In fact, the application of relation (173) (*i.e.* creating an intermediate layer of density σ_{k-1}) is not possible if the buoyancy is not a function of only one variable. Taking into account the joint effects of salinity and temperature introduces a degree of supplementary freedom. In this case, it is necessary to extract from the layer k a quantity of heat equal to the sum of the heat required to match the buoyancy value of the part above the slice of water entrained and of the heat corresponding to the increment $(b'_1 - b_1)$ of the mixed layer. In this framework, BLECK *et al.* (1992) have developed a particular method to schematize these transfers. Assume a surface layer of temperature T_1, of salinity S_1 and of thickness z_1. Let T_k, S_k, z_k be the corresponding characteristics for the layer k. If $(z_1 - L) > 0$, the procedure transfers the excess of the heat contained in a layer of thickness $(z_1 - L)$ to the layer k. Since we are in isopycnic coordinates, this transfer will be paired with a heat transfer of this ascending layer to the mixed layer. Let ΔT be the maximum allowed growth of the temperature of the mixed layer allowed during

a time step. We note by z the thickness of the fraction of the intermediate layer of vertical extension $(z_1 - L)$ which allows an augmentation of temperature ΔT. Since the mixed layer has a buoyancy greater than that of layer k, the problem becomes one of determining z. The heat transfer to layer k leads to a temperature T_{new} such that (figure 16) :

$$T_1 z + T_k z_k - \Delta T(z_1 - z) + T_{new}(z + z_k) \tag{174}$$

Let :

$$T_{new} = \frac{(T_1 + \Delta T)\, z - \Delta T z_1 + T_k z_k}{z + z_k} \tag{175}$$

and the salinity :

$$S_{new} = \frac{S_1 z + S_k z_k}{z + z_k} \tag{176}$$

These two characteristics must satisfy $\sigma(T_{new}, S_{new}) = \sigma_k$. In assuming an equation of state of third degree :

$$\sigma(T, S) = c_1 + c_2 T + c_3 S + c_4 T^2 + c_5 TS + c_6 T^3 + c_7 T^2 S \tag{177}$$

and in expressing T_{new} and S_{new} through the forms $(az + b)/(cz + d)$ and $(ez + f)/(cz + d)$, the conservation of σ_k requires solving the polynomial equation :

$$a_3 z^3 + a_2 z^2 + a_1 z + a_0 = 0 \tag{178}$$

The expressions for coefficients a_0, a_1, a_2, a_3 are given in appendix E of BLECK *et al.* (1992). Moreover, the introduction of a supplementary layer σ_{k-1} implies also the restoration of motion parameters. Now, the MICOM theory assumes that the exchanges of momentum occur immediately after the mass transfers between the mixed layer and sub-adjacent layers (BLECK *et al.* 1989). For a given layer, the momentum remains constant during the process of rearranging the sub-surface layers concerned in the evolution algorithm of the mixed layer.

9.3 Usage

In the MICOM code, the numerical calculation of the evolution of the surface mixed layer is carried out by the subroutine :

`subroutine mxlayr` (argument list)

9.3.1 Order of operations

Recall that, while deepening the mixed layer is relatively easy to model, to reproduce its recession is more complex. Assume a surface layer of characteristics T_1, S_1, z_1. In the original detrainment algorithm developed by BLECK *et al.* (1989) for one variable of state, later adapted by BLECK *et al.* (1992) for the two variables T and S, the heat acquired by the ocean surface during a time step is not distributed over a slice of water of thickness equal to the Monin-Obukov length L. In fact, we introduce the thickness h given by the relation (162) proposed by GASPAR (1988). So, the old mixed layer is divided into a new surface layer of characteristics T_1', S_1, h and a fossil layer of thickness $(z_1 - h)$. This last layer is also partitioned into two sub-layers :

1) a slice of water of density equal to that of the sub-adjacent layer σ_k and of salinity S_1 ;

2) an intermediate layer of temperature T_1' and of salinity S_1.

The option of parametrization of dissipation actually used in version 2.6 of MICOM is the formulation of GASPAR (1988). The lines of code of option 1 based on the method of KRAUS & TURNER (1967) are still available but commented out.

```
c
c --- determine turb.kin.energy generation due to wind stirring (ustar3) and
c --- diabatic forcing (buoyfl).  ustar3,buoyfl are computed in subr.  -thermf-
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
c --- option 1 :  k r a u s - t u r n e r mixed-layer t.k.e.  closure
c
ccc em=0.8*exp(-p(i,j,2)/(50.*onem)) !  hadley centre choice (orig.:  1.25)
ccc en=0.15 !  hadley centre choice (orig.:  0.4)
ccc thermg=.5*((en+1.)*buoyfl(i,j)+(en-1.)*abs(buoyfl(i,j)))
ccc turgen(i,j)=delt1*(2.*em*g*ustar3/thref+thknss*thermg)/thref
c
c --- find monin-obukhov length in case of receding mixed layer (turgen < 0).
c --- the monin-obukhov length is found by stipulating turgen = 0.
c --- store temporarily in 'sdot'.
c
ccc if (turgen(i,j).lt.0.)  then
ccc sdot(i,j)=-2.*em*g*ustar3/min(-epsil,thref*thermg)
ccc else
ccc sdot(i,j)=thknss
ccc end if
c
c --- option 2 :  g a s p a r mixed-layer t.k.e.  closure
c
      dpth=thknss/onecm
      ekminv=abs(corio(i,j))/max(epsil,ustar(i,j))
      obuinv=-buoyfl(i,j)/max(epsil,ustar3)
      ex=exp(min(50.,dpth*obuinv))
      alf1=ea1+ea2*max(1.,2.5*dpth*ekminv)*ex
      alf2=ea1+ea2*ex
      cp1=((1.-em5)*(alf1/alf2)+.5*em4)*athird
      cp3=(em4*(em2+em3)-(alf1/alf2)*(em2+em3-em3*em5))*athird
      ape=cp3*ustar3/thref+cp1*dpth*buoyfl(i,j)
c
```

Once we have calculated A_p with relation (148), we carry out the test :

> if $A_p > 0$, the TKE variation ($E\Delta t$) of the mixed layer during a time step is positive. To maintain the equilibrium formulated by the equation (140), this growth represents the TKE that will be consumed by entrainment into the mixed layer. It is calculated from the form (147). A first inference of the new thickness z_1' of the mixed layer is given by z_1;

if $A_p < 0$, the TKE variation is negative. The TKE contained in the mixed layer will therefore diminish by a quantity $E\Delta t = A_p$. A first inference of the new thickness z_1' of the mixed layer is given by the relation (162) under the condition that it be less than z_1.

```
c
      if(ape.lt.0.)  then                                  !detrainment
      turgen(i,j)=(g*delt1/thref)*ape
      sdot(i,j)=min(thknss,g*cp3/(cp1*max(epsil,obuinv)))
c
      else                                                !entrainment
      cc4=2.*em4/(em1*em1) * alf1*alf1
      spe=(em2+em3)*ustar3/thref+0.5*dpth*buoyfl(i,j)
      turgen(i,j)=(g*delt1/thref)*(sqrt((.5*ape-cp1*spe)**2
     .  +2.*cc4*ape*spe)-(.5*ape+cp1*spe))/(cc4-cp1)
      sdot(i,j)=thknss
      end if
c
```

Then, we calculate the form (169) iterating over k as long as the value of h obtained remains greater than that of the upper interface of the k^{th} layer situated at the position z_{k-1}. When this condition is no longer true, we obtain the order l of the relation (169) and as a consequence, a second inference of z_1'.

```
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
c --- util1,util2 are used to evaluate pot.energy changes during entrainment
      util1(i,j)=thmix(i,j,n)*thknss
   86 util2(i,j)=thmix(i,j,n)*thknss**2
c
c --- find pnew in case of mixed layer deepening (turgen > 0).  store in 'sdot'.
c --- entrain as many layers as needed to deplete -turgen-.
c
      do 85 k=2,kk
      kn=k+nn
      do 85 i=ifp(j,l),ilp(j,l)
      pnew=(2.*turgen(i,j)+theta(k)*p(i,j,k)**2-util2(i,j))/
     .  max(epsil,theta(k)*p(i,j,k) -util1(i,j))
c --- stop iterating for 'pnew' as soon as pnew < k-th interface pressure
      if (pnew.lt.p(i,j,k)) pnew=sdot(i,j)
c --- substitute 'pnew' for monin-obukhov length if mixed layer is deepening
      if (turgen(i,j).ge.0.)  sdot(i,j)=pnew
c
      util1(i,j)=util1(i,j)+theta(k)*dp(i,j,kn)
   85 util2(i,j)=util2(i,j)+theta(k)*(p(i,j,k+1)**2-p(i,j,k)**2)
```

```
c
```

Before validating this first result, we require the numerical value which will be retained to be located between the bottom and a minimum thickness value of the mixed layer.

```
c
      do 42 i=ifp(j,l),ilp(j,l)
c --- store (pnew - pold) in 'sdot'.
c --- don't allow mixed layer to get too deep or too shallow.
      sdot(i,j)=min(p(i,j,kk+1),max(thkmin*onem,sdot(i,j)))
     .  -dp(i,j,k1n)
      klist(i,j)=2
      tdp(i,j)=0.
   42 sdp(i,j)=0.
c
      thknss=dp(i,j,k1n)
      pnew=thknss+sdot(i,j)
c
```

After this, the sign of the variation $\Delta z = z'_1 - z_1$ is used as a test :

if $\Delta z > 0$, we confirm the result $h_s = z'_1$ and we calculate the new values of temperature, salinity, and density of the mixed layer ;

```
c
c --- (mixed layer d e e p e n s)
c
      saln(i,j,k1n)=(saln(i,j,k1n)*thknss+sdp(i,j))/pnew
      temp(i,j,k1n)=(temp(i,j,k1n)*thknss+tdp(i,j))/pnew
      dp(i,j,k1n)=pnew
      thmix(i,j,n)=sig(temp(i,j,k1n),saln(i,j,k1n))-thbase
c
```

if $\Delta z < 0$, we proceed in two steps :

1) from the general equation of internal energy (124), we determine the growth of temperature ΔT undergone by a fictitious layer of thickness z'_1 given by the relation (162) from the theory of GASPAR (1988).

2) for the mixed layer to maintain this thickness z'_1, we conserve the salt content (see figure 18). Then, we apply the relation of state giving the temperature T_{new} as a function of σ_k (which is conserved) and of salinity S_{new} which we want to calculate. The growth Δz of the thickness of the sub-adjacent layer involves a heat transfer in the fictitious surface layer which should be positive. (see figure 19).

```
c
```

```
      k=klist(i,j)
      kn=k+nn
      dpkn=max(dp(i,j,kn),0.)
      skn=saln(i,j,kn)
      tkn=temp(i,j,kn)
      skn1=saln(i,j,k1n)
      tkn1=temp(i,j,k1n)
c
c - assume that incoming heat is distributed over Monin-Obukhov depth l.
c - 'dtemp' is the resulting mixed-layer temperature increment.  split
c - mixed layer below l into one part of depth z cooled and detrained
c - into layer k, and a part heated to match temperature rise above l.
c - to balance thermal energy while maintaining reference density in
c - layer k, z must satisfy cc3*z**3 + cc2*z**2 + cc1*z + cc0 = 0.
c
      dtemp=surflx(i,j)*delt1*sdot(i,j)*g/(spcifh*thknss*pnew)
c
c - compare dtemp with heating rate 'dtmx' resulting
c - from 100% detrainment
c
      snew=(skn*dpkn-skn1*sdot(i,j))/(dpkn-sdot(i,j))
      tnew=tofsig(theta(k)+thbase,snew)
      dtmx=((tkn-tnew)*dpkn-(tkn1-tnew)*sdot(i,j))/pnew
c
```

At this stage, it is therefore possible to compare this heat gain taken to z_1' (which translates to an increase of the surface temperature dT) with the temperature growth ΔT found before :

a) if $dT \leq \Delta T$, in this configuration, 100% detrainment is possible. We attribute the new values T_{new} and S_{new} to layer k then, concerning the mixed layer, we put $T_1' = T_1 + dT$ and $z'1 = z1 + \Delta z$. In operating this way we guard against the non-representative surface heating because, unless we have obtained exactly $dT = \Delta T$, the depth of the mixed layer which we fix is greater than what it should be. In all cases, it should be greater than the threshold value which was fixed by the run. The surface density becomes $\sigma_1' = \sigma(T_1', S_1)$

```
c
c --- 100% detrainment possible
c
      dp(i,j,k1n)=pnew
      dp(i,j,kn) =dpkn-sdot(i,j)
      temp(i,j,k1n)=tkn1+dtmx
      temp(i,j,kn)=tnew
      saln(i,j,kn)=snew
c
```

b) if $dT > \Delta T$, we distribute the heat over a surface layer of temperature $T_1'' = T_1 + dT$ and of thickness z_1'' such that $z_1'' > z_1'$ (see figure 20). The sub-adjacent layer k is also going to thicken and the new characteristics T_{new} and S_{new} of this layer should also satisfy :

$$\sigma(T_{new}, S_{new}) = \sigma_k \tag{179}$$

In keeping with the preceding case, partial detrainment is now possible. The density of the mixed layer now becomes $\sigma_1'' = \sigma(T_1'', S_1)$. The new characteristics of layer k have particular forms given by appendix E of BLECK *et al.* (1992), the conservation of σ_k comes from the solution of a polynomial of the form (178).

```
c
c --- partial detrainment only.
c --- new (t,s) in layer k will be t=(a*z+b)/(z+d), s=(e*z+f)/(z+d).
      a=tkn1+dtemp
      b=(tkn*dpkn-dtemp*thknss)/onem
      d=dpkn/onem
      e=skn1
      f=skn*dpkn/onem
c
      c1msig=c1-1.e3*(theta(k)+thbase)
      cc0=d*d*(d*c1msig+b*c2+f*c3)+b*(d*f*c5+b*(d*c4+b*c6+f*c7))
      cc3= ( c1msig+a*c2+e*c3)+a*( e*c5+a*( c4+a*c6+e*c7))
      cc1=d*(3.  *d*c1msig+(2.*b +a*d)*c2+(2.  *f+d*e)*c3)+b*((2.*a*d
     .  +b )*c4+3.*a*b*c6+(2.*a*f+b*e)*c7)+(a*d*f+b*(d*e+ f))*c5
      cc2= (3.  *d*c1msig+(2.*a*d+b )*c2+(2.*d*e+ f)*c3)+a*((2.*b
     .  +a*d)*c4+3.*a*b*c6+(2.*b*e+a*f)*c7)+(b *e+a*( f+d*e))*c5
c
      if (ccubq(x)**3+ccubr(x)**2.gt.0.)  then
c --- one real root
      num=1
      z=root(x)
      else
c --- three real roots
      num=3
      z=root1(x)
      end if
c
c --- does root fall into appropriate range?
      if (z.lt.-.001.or..99*z*onem.gt.-sdot(i,j)) then
      work(1)=z
      if (num.eq.3) then
      work(2)=root2(x)
      work(3)=root3(x)
```

```
            end if
      c
      c --- detrain amount 'z' into layer k c
            sdot(i,j)=max(sdot(i,j),-z*onem)
            dp(i,j,k1n)=thknss+sdot(i,j)
            dp(i,j,kn) =dpkn -sdot(i,j)
            temp(i,j,k1n)=tkn1+dtemp
            temp(i,j,kn)=(a*z+b)/(z+d)
            saln(i,j,kn)=(e*z+f)/(z+d)
      c
```

The specific volume of the surface layer is given by the equation of state.

```
c
      thmix(i,j,n)=sig(temp(i,j,k1n),skn1)-thbase c
```

The following operation consists of accounting for the thickness variation of the surface layer.

```
c
c --- store 'old' interface pressures in -pu,pv-
c
      do 882 k=2,kk+1
      do 882 j=1,jj1
c
      do 881 l=1,isu(j)
      do 881 i=ifu(j,l),ilu(j,l)
  881 pu(i,j,k)=min(depthu(i,j),.5*(p(i,j,k)+p(i-1,j,k)))
c
      do 882 l=1,isv(j)
      do 882 i=ifv(j,l),ilv(j,l)
  882 pv(i,j,k)=min(depthv(i,j),.5*(p(i,j,k)+p(i,j-1,k)))
c
c --- store 'new' layer thicknesses in -dpu,dpv-
c
      do 883 j=1,jj1
      do 883 k=1,kk
      do 883 l=1,isp(j)
      do 883 i=ifp(j,l),ilp(j,l)
  883 p(i,j,k+1)=p(i,j,k)+dp(i,j,k+nn)
c
      do 834 k=1,kk
      kn=k+nn
      do 834 j=1,jj1
c
      do 831 l=1,isu(j)
      do 831 i=ifu(j,l),ilu(j,l)
```

```
  831 dpu(i,j,kn)=max(0.,
     .  min(depthu(i,j),.5*(p(i,j,k+1)+p(i-1,j,k+1)))-
     .  min(depthu(i,j),.5*(p(i,j,k )+p(i-1,j,k ))))
c
      do 834 l=1,isv(j)
      do 834 i=ifv(j,l),ilv(j,l)
  834 dpv(i,j,kn)=max(0.,
     .  min(depthv(i,j),.5*(p(i,j,k+1)+p(i,j-1,k+1)))-
     .  min(depthv(i,j),.5*(p(i,j,k )+p(i,j-1,k ))))
c
```

The last step of the calculation consists of the redistribution of momentum over the water column, accounting for the possibility of momentum mixing by diffusion.

```
c
c --- redistribute momentum in the vertical.
c --- homogenize (u,v) over depth range defined in -util1,util2-
c
c --- thk>0 activates momentum diffusion across mixed-layer interface
      thk=vertmx*onecm*delt1
c
      do 97 j=1,jj1
c
      do 83 l=1,isu(j)
c
      do 822 i=ifu(j,l),ilu(j,l)
      util1(i,j)=max(dpu(i,j,k1n),pu(i,j,2)+thk)
      uflux(i,j)=0.
  822 util3(i,j)=0.
c
      do 82 k=1,kk
      do 82 i=ifu(j,l),ilu(j,l)
      delp=max(0.,min(util1(i,j),pu(i,j,k+1))
     .  -min(util1(i,j),pu(i,j,k )))
      uflux(i,j)=uflux(i,j)+u(i,j,k+nn)*delp
   82 util3(i,j)=util3(i,j) +delp
c
      do 83 i=ifu(j,l),ilu(j,l)
   83 u(i,j,k1n)=uflux(i,j)/util3(i,j)
c
      do 84 l=1,isv(j)
c
      do 844 i=ifv(j,l),ilv(j,l)
      util2(i,j)=max(dpv(i,j,k1n),pv(i,j,2)+thk)
      vflux(i,j)=0.
   844 util4(i,j)=0.
c
```

```
      do 80 k=1,kk
      do 80 i=ifv(j,l),ilv(j,l)
      delp=max(0.,min(util2(i,j),pv(i,j,k+1))
     .  -min(util2(i,j),pv(i,j,k )))
      vflux(i,j)=vflux(i,j)+v(i,j,k+nn)*delp
   80 util4(i,j)=util4(i,j) +delp
c
      do 84 i=ifv(j,l),ilv(j,l)
   84 v(i,j,k1n)=vflux(i,j)/util4(i,j)
c
      do 97 k=2,kk
      kn=k+nn
c
      do 96 l=1,isu(j)
      do 96 i=ifu(j,l),ilu(j,l)
      pu(i,j,k)=pu(i,j,k-1)+dpu(i,j,kn-1)
      q=max(0.,min(1.,(util1(i,j)-pu(i,j,k))/(dpu(i,j,kn)+epsil)))
   96 u(i,j,kn)=u(i,j,k1n)*q+u(i,j,kn)*(1.-q)
c
      do 97 l=1,isv(j)
      do 97 i=ifv(j,l),ilv(j,l)
      pv(i,j,k)=pv(i,j,k-1)+dpv(i,j,kn-1)
      pv(i,j,k)=pv(i,j,k-1)+dpv(i,j,kn-1)
      q=max(0.,min(1.,(util2(i,j)-pv(i,j,k))/(dpv(i,j,kn)+epsil)))
   97 v(i,j,kn)=v(i,j,k1n)*q+v(i,j,kn)*(1.-q)
```

9.3.2 Flowchart

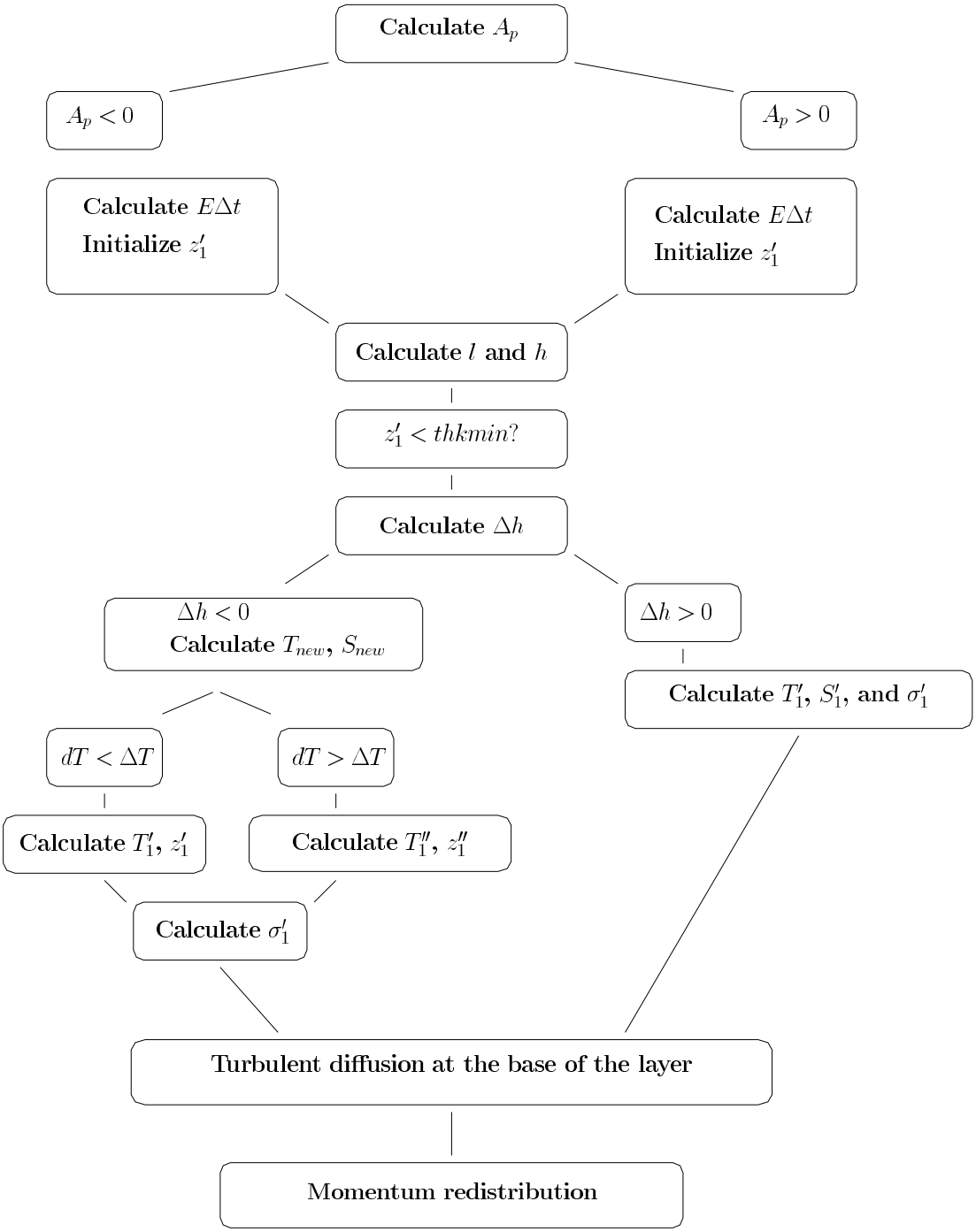

Figure 17: *Flowchart of the calculation of the mixed layer evolution in MICOM 2.6*

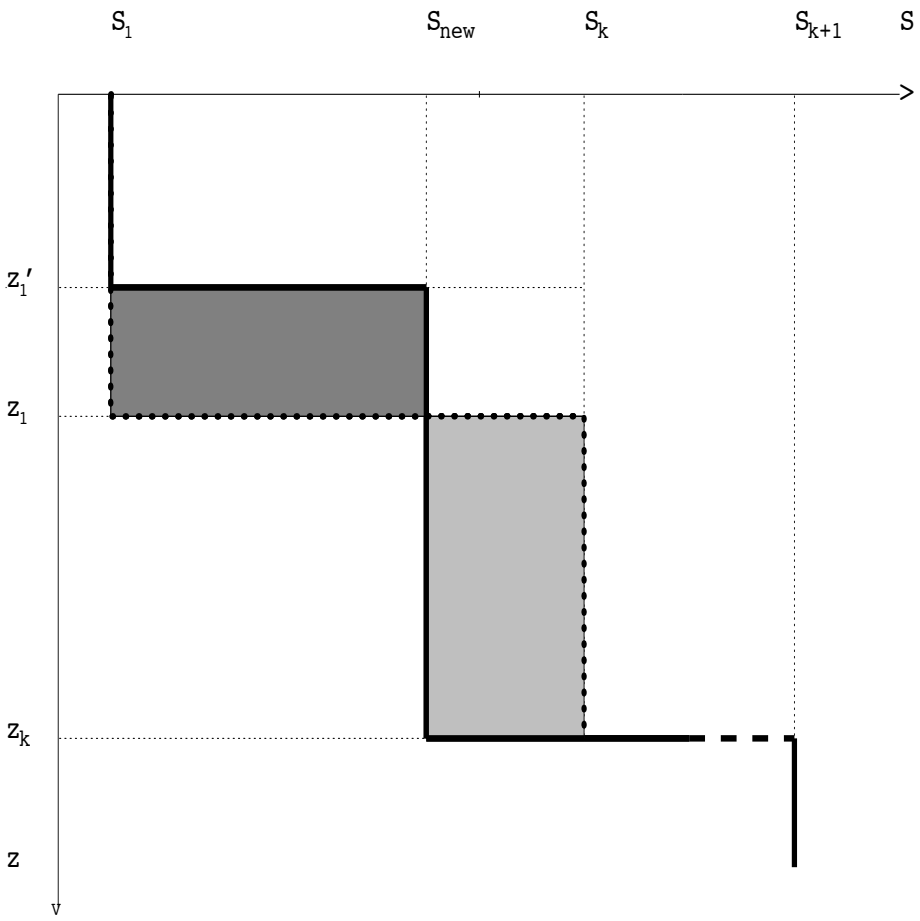

Figure 18: *Conservation of salt in updating the mixed layer.*

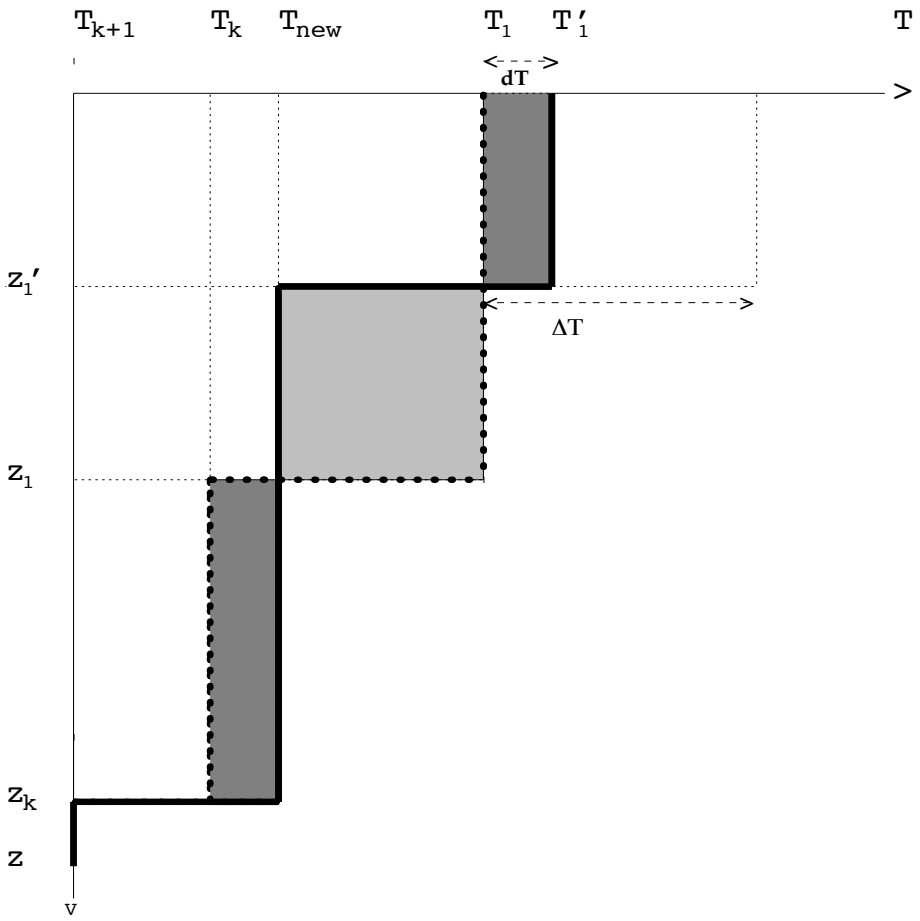

Figure 19: *Illustration of the mechanism of updating the mixed layer in the case when 100% detrainment is possible.*

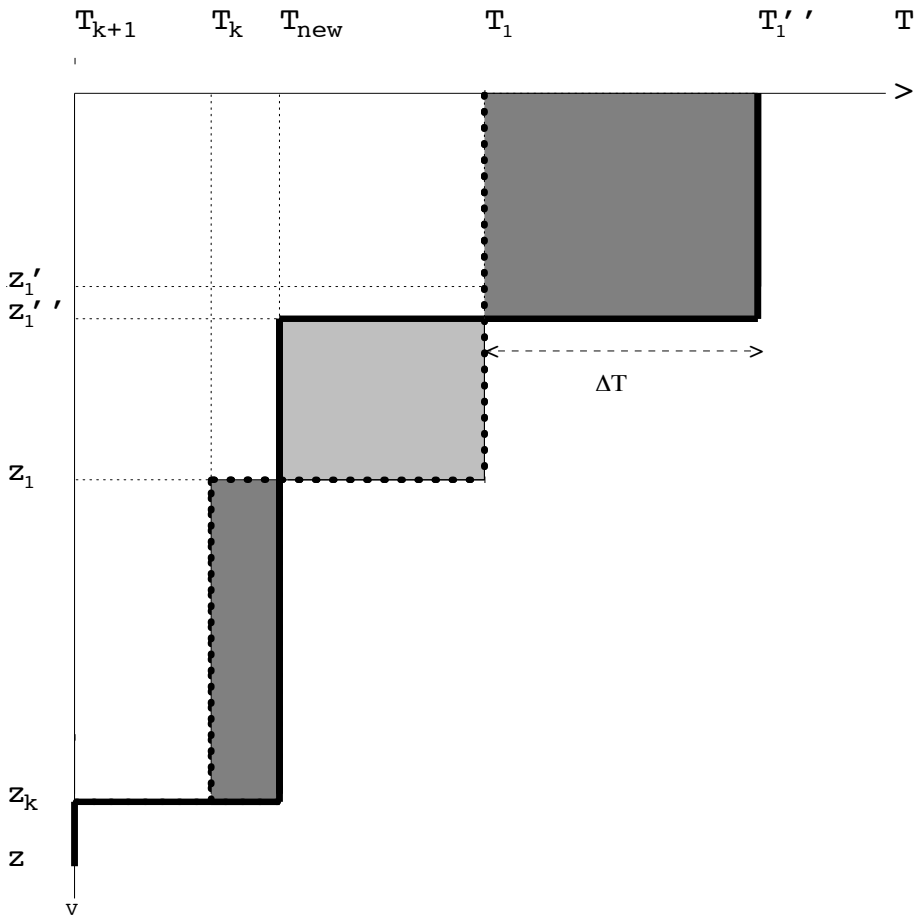

Figure 20: *Illustration of the mechanism of updating the mixed layer in the case when partial detrainment is possible.*

9.4 Variables

9.4.1 Identification

Notation in the theory of GASPAR (1988)	Notation in **mxlayr.f**
h/l	`alf1`
h/l_p	`alf2`
A_p, S_p	`ape,cpe`
B	`buoyfl(i,j)`
C_4	`cc4`
C_{p1}, C_{p3}	`cp1,cp3`
h	`dpth`
a_1, a_2	`ea1,ea2`
$1/\lambda$	`ekminv`
$m_1, ..., m_5$	`em1,...,em5`
$\exp(h/L)$	`ex`
$1/L$	`obuinv`
$E\Delta t$	`turgen(i,j)`
u_*	`ustar(i,j)`

Notation in the theory of BLECK *et al.* (1992)	Notation in **mxlayr.f**
a, b, c, d, e, f	`a,b,c,d,e,f`
a_0, a_1, a_2, a_3	`cc0,cc1,cc2,cc3`
$c_1, c_2, c_3, c_4, c_5, c_6, c_7$	`c1,c2,c3,c4,c5,c7`
$d(c_1 - \sigma_k)$	`c1msig`

9.4.2 Global variables

integer variables : `klist(i,j)`

real variables : `buoyfl(i,j)`, `delt1`, `depthu(i,j)`, `depthv(i,j)`, `dp(i,j,k)`, `dpu(i,j,k)`, `dpv(i,j,k)`, `g`, `onecm`, `p(i,j,k)`, `pu(i,j,k)`, `pv(i,j,k)`, `saln(i,j,k)`, `sdot(i,j)`, `tdp(i,j)`, `temp(i,j,k)`, `thkmin`, `thmix(i,j,k)`, `tracer(i,j,k)`, `turgen(i,j)`, `u(i,j,k)`, `v(i,j,k)`, `uflux(i,j)`, `vflux(i,j)`, `ustar(i,j)` `util1(i,j)`, `util2(i,j)`, `util3(i,j)`, `util4(i,j)`, `vertmx`

logical variables : `thermo`, `trcout`

9.4.3 Local variables

a,b,c,d,e,f	coefficients in the expressions of new characteristics T and S of layer k
alf1	stability parameter h/l
alf2	stability parameter in the absence of rotation h/l_p
ape	term in the expression of entrainment
cc4	intermediate coefficient parametrizing turbulent effects
cp1,cp3	coefficients in the expression of the entrainment
c1msig	coefficient in the calculation of zeroes of the polynomial giving the new characteristics of the layer sub-adjacent to the mixed layer
delp	pressure variation at interfaces
dpth	transformation of the pressure difference in the mixed layer in thickness units (cm)
dtemp	variation ΔT of temperature in the mixed layer
dtmx	temperature increment dT corresponding to the buoyancy transfer contained in the slice of water of thickness Δz
dpkn	mixed layer thickness
ekminv	inverse of the Ekman length
em1,...,em5	coefficients of the turbulent effects parametrization
ex	exponential of the Monin-Obukov stability parameter h/L
num	number of real roots
obuinv	inverse of the Monin-Obukov length
pnew	mixed layer thickness (cm)
q	relative variation of momentum in the mixed layer due to its deepening
skn,skn1	salinity of the mixed layer at two consecutive instants
snew	salinity of the mixed layer
spe	term in the expression of the entrainment
thk	augmentation of mixed layer thickness by turbulent diffusion
thknss	pressure difference in the mixed layer (mb) at $t = n\Delta t$ (s)
tkn,tkn1	temperatures of the mixed layer at two consecutive instants
tnew	temperature of the mixed layer

`ustar3`	`ustar(i,j)**3`
`work(3)`	temporary storage array
`z`	real root of the polynomial

9.5 Procedures

Functions	`harmon`, `ccubq`, `ccubr`, `ccubqr`, `ccubs1`, `ccubs2`, `root`, `ccubrl`, `ccubim`, `root1`, `root2`, `root3`
Subroutines	`prtij`, `advem`

10 Ocean-atmosphere exchanges : thermf.f

In MICOM 2.6, the ocean-atmosphere exchanges considered are of three types :

- The radiative exchanges R : balance of incident solar radiation and radiation emitted by the sea surface.
- Turbulent heat transfers
  1. A latent heat transfer $\mathcal{E}$ due to evaporation of seawater ;
  2. A sensible heat transfer $\mathcal{H}$ which is established by convection when a significant difference exists between the temperature of the sea surface and that of the air.
- Mechanical energy transfers : essentially the effect of wind

10.1 Formalism and numerical techniques

In the field of applications of MICOM 2.6, the atmospheric forcings are integrated at each point *via* the following files of thermodynamic climatological parameters :

. radiation : `radflx(i,j,l)` ;
. wind at the sea surface (10m) : `wndspd(i,j,l)` ;
. air temperature : `airtmp(i,j,l)` ;
. water vapor content : `vapmix(i,j,l)` ;
. precipitation : `precip(i,j,l)`.

The numerical values furnished at times $l\Delta\mathcal{T}$ where $\Delta\mathcal{T}$ is the sampling period are interpolated to the simulation's timestep, $n\Delta t$, using the coefficients w_0, w_1, w_2, w_3.

10.1.1 Heat balance

In MICOM 2.6, the effect of ocean-atmosphere exchanges (except the mechanical effect of the wind) on the mixed layer is summed in the calculation of thermal balance :

$$B = R + \mathcal{H} + \mathcal{E} \tag{180}$$

A positive (*i.e.*, upward) flux of sensible heat corresponds to a loss, a contribution of energy by the sea following the sign of the difference between the sea and the atmospheric boundary layer :

$$\mathcal{H} = C_{p_{air}} E_x \, (T_s - T_a) \tag{181}$$

E_x is an exchange coefficient such that : $E_x = \rho_a C_T W$

ρ_a : mass/volume of the air
C_T : heat transfer coefficient
$C_{p_{air}}$: specific heat of the air
T_s : sea surface temperature
T_a : temperature in the atmospheric boundary layer
W : wind velocity

Responsible for important quantities of heat exchanged between the sea and atmosphere by evaporation, the latent heat flux always induces a heat loss by the ocean. To formulate this term in the balance estimation, in MICOM 2.6, we use the expression :

$$\mathcal{E} = E_x \mathcal{L}(H_u - E_v) \tag{182}$$

With :

$\mathcal{L}$: latent heat of vaporization
H_u : specific humidity
E_v : evaporation

Moreover, we account for the precipitation-evaporation balance in the evolution of the salinity (therefore the density) of the mixed layer by prescribing the precipitation.

10.1.2 Mechanical energy transfers

A wind representable by the vector $\mathbf{W}(t)$ induces on the surface a wind stress $\boldsymbol{\tau}_s(t)$ which can be well-expressed by a quadratic expression of type :

$$\boldsymbol{\tau}_s = \rho_a \; C_D \; |\mathbf{W}|\mathbf{W} \tag{183}$$

C_D is a coefficient realizing sea surface drag. We put $C_D \simeq 10^{-3}$. The expression of the drag velocity at the surface u_* is obtained by the relation :

$$u_*^2 \; \mathbf{x} = \boldsymbol{\tau}_s / \rho_0 \tag{184}$$

At the point z, we more often use the classic parametrization of the concept of turbulent diffusion :

$$\boldsymbol{\tau}(z) = \rho \; K_M \; \frac{\partial \mathbf{u}}{\partial z} \tag{185}$$

with :

$\mathbf{x}$: horizontal unit vector
K_M : coefficient of turbulent momentum diffusion

10.2 Usage

10.2.1 Order of operations

First, we interpolate the values of thermodynamic parameters (radiation, wind, *etc*) at an instant in the simulation. From there, it is possible to determine the flux of latent and sensible heat as well as the velocity drag at the surface. The following step accounts for the thermal balance by proceeding to the inference of new values of temperature, salinity, and density of the mixed layer. During the last phase, we calculate the buoyancy growth necessary to establish the Monin-Obukov length which is the reference in the step of modeling the evolution of the surface layer (*cf.* § 9).

10.2.2 Flowchart

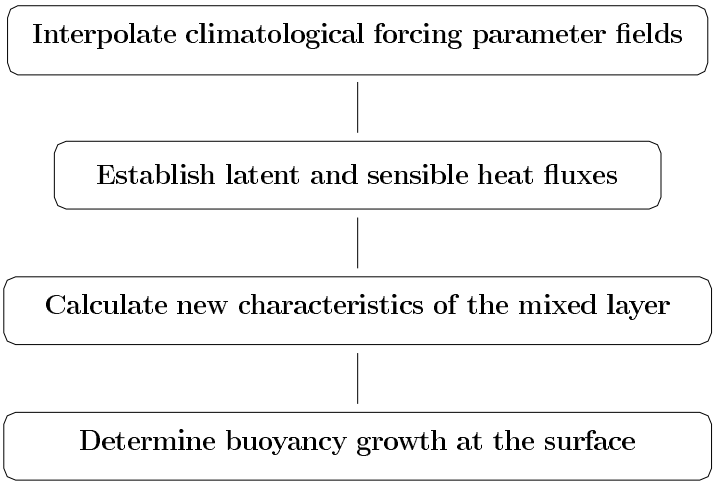

Figure 21: *Order of the thermal balance calculation in the mixed layer in MICOM 2.6*

10.3 Variables

10.3.1 Identification

Notation in the theory	Notation in **thermf.f**
B	`surflx(i,j)`
$C_{p_{air}}$	`csubp`
$C_{p_{water}}$	`spcifh`
C_T	`ct`
E_x	`exchng`
E_v	`vpmx`
$\mathcal{E}$	`evap`
H_u	`qsatur`
$\mathcal{L}$	`evaplh`
R	`radflx`
u_*	`ustar`
W	`wind`
ρ_a	`airdns`

10.3.2 Global variables

integer variables : `ifp(i,l)`, `ilp(i,l)`, `jj1`

real variables : `buoyfl(i,j)`, `delt1`, `dp(i,j,k)`, `saln(i,j,k)`, `surflx(i,j)`, `temp(i,j,k`, `thmix(i,j,n)`, `thref`, `ustar(i,j)`

logical variables : `diagno`

10.3.3 Local variables

`airt`	air temperature
`emnp`	evaporation-precipitation balance
`empcum`	cumulative balance of evaporation-precipitation
`evap`	evaporation balance
`exchng`	coefficient of heat transfer
`j,l`	intermediate indices
`prcp`	precipitation
`radfl`	radiative flux
`thknss`	mixed layer thickness
`vpmx`	evaporation
`watts`	cumulative flux
`wind`	wind

10.4 Procedures

Functions `dsigds`, `dsigdt`, `qsatur`

11 Calculational grid

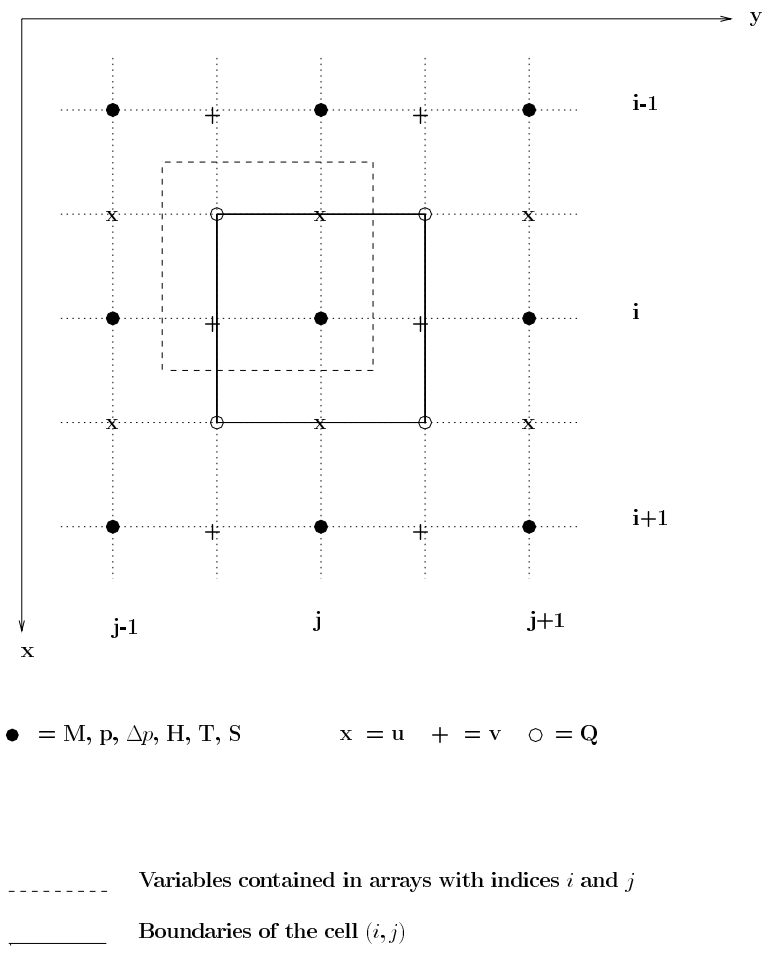

Figure 22: *Distribution of variables on an Arakawa C grid*

12 Equation of state

Rather than do the numerical calculations as a function of density, which comes into the analytic formulations by its inverse, we introduce the specific volume α such that :

$$\alpha = \frac{1}{\rho} \tag{186}$$

If on the other hand, we consider the variable σ such that :

$$\sigma = \frac{\rho}{\rho_0} - 1 \tag{187}$$

we then obtain :

$$\alpha = \frac{1}{\rho_0(1+\sigma)} \approx \frac{1}{\rho_0}(1-\sigma) \tag{188}$$

The system of units used in MICOM is the CGS system. Consider a layer of density $\rho_k = 1.025\ g/cm^3$. Using a reference $\rho_0 = 1\ g/cm^3$, we then have $\sigma_k \simeq 0,025$. In the code, the authors have introduced notations in comments in the file **blkdat.f.** :

```
c     alpha=1/rho=thref*(1.-theta(k))
```

and we use `thref=1.` The numerical values of the variable `theta(k)` correspond therefore (by a factor near 10^{-3}) to those we habitually use to describe a field of oceanic density through the variable $\boldsymbol{\sigma}_T$ which we introduce by the relation :

$$d = 1 + 10^{-3}\boldsymbol{\sigma}_T \tag{189}$$

and where we then have $d \simeq 1$. Moreover, it is then possible to express the variables representing the density by a polynomial function of the temperature (`t` in ^oC) and of the salinity (`s` in $^o/_{oo}$). In MICOM, we use the equation advocated by FRIEDRICH & LEVITUS (1972) :

```
sig(t,s)=(c1+c3*s+t*(c2+c5*s+t*(c4+c7*s+c6*t)))*1.e-3
```

with : $c1 = -7.2169\ 10^{-2}$; $c2 = 4.9762\ 10^{-2}$; $c3 = 8.0560\ 10^{-1}$; $c4 = 7.5911\ 10^{-3}$; $c5 = -3.0063\ 10^{-3}$; $c6 = 3.5187\ 10^{-5}$ and $c7 = 3.7297\ 10^{-5}$. The partial derivatives of this expression with respect to the temperature and the salinity give the relations :

```
dsigdt(t,s)=(c2+c5*s+2.*t*(c4+c7*s+1.5*c6*t))*1.e-3

dsigds(t,s)=(c3+t*(c5+t*c7))*1.e-3
```

The temperature as a function of specific volume (`r`) and of the salinity are then given by the formula :

```
tofsig(r,s)=-cubrl(r,s)+sqrt(3.)*cubim(r,s)-athird*a2(s)
```

with the internal functions :

```
cubrl(r,s)=sqrt(-cubq(s))*cos(cuban(r,s))
cubq(s)=athird*a1(s)-(athird*a2(s))**2
cuban(r,s)=athird*atan2(sqrt(amax1(0.,-(cubq(s)**3+cubr(r,s)**2))),cubr(r,s))
cubr(r,s)=athird*(.5*a1(s)*a2(s)-1.5*(a0(s)-1.e3*r/c6))-(athird*a2(s))**3
cubim(r,s)=sqrt(-cubq(s))*sin(cuban(r,s))
a0(s)=(c1+c3*s)/c6
a1(s)=(c2+c5*s)/c6
a2(s)=(c4+c7*s)/c6
```

and the parameter : `athird=1./3.`

Note : All modification of the preceding expression of the equation of state imply changing the internal functions declared in the head of the sub-program `mxlayr`

13 Sub-programs

13.1 Functions

Calculation of the harmonic average of two variables `a` and `b` :

```
harmon(a,b)=2.*a*b/(a+b)
```

Calculation of the latitude `alat` as a function of the distance `dist1` to the equator and of the size of the mesh `grid` in degrees of latitude :

```
alat(dist1,grid)=(2.*atan(exp(dist1*grid/radian))-pi/2.)
```

Calculation of the distance `dist` to the equator as a function of the latitude `alat` and of the size of the mesh `grid` in degrees of latitude :

```
dist(alat1,grid)=alog(tan((2.*alat1+pi)/4.))*radian/grid
```

Determination of the depth at calculational points u and v :

```
uvdep(a,b)=min(a,b)
```

13.2 Subroutines

```
subroutine prtij (imid,jmid,k,u,scu,v,scv,dp,scp,t,sct,s,scs)
```

From a given layer `k`, this subroutine prints a window of $5*5$ mesh points around a point which has coordinates identified by the first two arguments : `imid` and `jmid`. The concerned variables are successively : `u`,`v`,`dp`,`t` and `s`. The output values are integers. The appropriate scaling factors `scu`, `scv`, `scp`, `sct` and `scs` should therefore be specified as input arguments.

14 MICOM notes

(1) The real-basin version of the Miami Isopycnic Coordinate Ocean Model (originally called ATLMIX, but now referred to as MICOM) contains a Kraus- Turner mixed layer, accommodates 2 independent thermodynamic variables (T,S), and supports variable bathymetry and irregular coastlines. The model is documented in BLECK/ROOTH/HU/SMITH (1992) (BRHS). The diapycnic mixing scheme, which didn't make it into that paper, is documented in a LaTeX file called DIAPYC.TEX.

(2) Model versions 1.x are snapshots taken at various stages of model development as bugs were discovered and loose ends were tied up. In versions 1.x, temperature and salinity are advected and diffused independently throughout the domain. Model versions 2.x treat temperature as a diagnostic variable in isopycnic layers (but not in the mixed layer). This cuts advection time in half and eliminates the need for a coordinate maintenance ("cabbeling") algorithm. Detrainment is patterned after Appendix E in BRHS. Versions 2.2 and higher include diapycnal mixing of buoyancy and salinity, with temperature in isopycnic layers inferred from the equation of state. Starting with version 2.3, the thermal forcing functions "T-hat" and Bowen ratio used in older versions are replaced by actual atmospheric temperature and humidity.

(3) Horizontal array dimensions are uniformly set to (IDM,JDM), even though most fields do not make use of the full allotted space due to horizontal staggering considerations (Arakawa C grid). Specifically, there are only (IDM-1)x(JDM-1) mass points, (IDM)x(JDM-1) U-velocity points and (IDM-1)x (JDM) V-velocity points. U and V points addressed as (I,J) are actually located at grid locations (I-1/2,J) and (I,J-1/2), respectively, relative to mass point (I,J). Vorticity and Coriolis parameter addressed as (I,J) are at (I-1/2,J-1/2). The distribution of water and land within the rectangle spanned by (IDM-1,JDM-1) is defined by the DEPTHS array. A zero depth means land.

(4) The two time slots required for all prognostic and some diagnostic variables are incorporated into the third (vertical) array dimension. The vertical grid index KM [as in TEMP(I,J,KM)] stands for model level K, time level M (as in "mid" time). Letter N [as in TEMP(I,J,KN)] indicates the "new" and "old" time level.

(5) We recommend that the code be compiled with all variables preset to negative-indefinite. On Cray machines, this is accomplished by compiling with the "-ei" option and loading with the "-f indef" option. Grid values whose use in the various finite-difference operators is anticipated (and only those) are set to zero during model initialization. This provides some safeguard against contamination by "spurious" land data in case of unforeseen complexities in coastline shape.

(6) If compiled with the "-ei" compiler option, model version 1 will crash upon encountering single-grid-interval coastal inlets or channels. Versions 2.1 - 2.3 contain code (16 lines starting with statement label 16 in subroutine BIGRID) to detect and fill in such features in the DEPTHS array. Version 2.4 does away with this restriction, i.e., allows single-point inlets (provided the above-mentioned 16 lines in BIGRID are removed). Single-point promontories and islands are okay.

(7) There are no restrictions regarding the number of islands, but the parameter MS must

be adjusted to the maximum number of "interruptions" of any grid row or column by land. Specifically, MS-1 interruptions are allowed. Interruptions in the diagonal direction (of interest to the Poisson solver) are governed by the parameter MSD. If MS and/or MSD are chosen too small, the program will stop in one of the routines BIGRID,INDXI,INDXJ during the initialization phase with an error message.

(8) Loops extending over the horizontal grid domain are triply nested to suspend arithmetic operations wherever a grid row crosses land. Innermost loop bounds are stored in arrays with names like IFP,ILP,IFU,ILU,JFV,JLV where the letters F and L stand for first and last grid point in a segment (segments are defined as contiguous 1-dimensional sets of "water" points), the letters P,U,V indicate whether the loop goes over mass (P) or velocity (U,V) points, and the letters I,J indicate whether the inner loop index is I or J. The middle loop extends over the number of segments in the grid row defined in the outer loop; this number is given in arrays named ISP,JSV, etc.

(9) The model does not use a rigid lid; instead, it advances the barotropic and baroclinic solutions using a split-explicit scheme. The barotropic streamfunction is therefore computed for diagnostic purposes alone. If the iteration count in the Poisson solver is set too low for the solution to converge, the model run itself will not be affected. No attempt is presently made to determine the correct streamfunction value along island boundaries. We suggest the following quick fix to prevent the Poisson solver from setting the streamfunction to zero along island boundaries (which it presently does) :

(a) Change the name of the GINDEX Common block in Subr. POISND to GINDX1.
(b) Make a copy of Subr. BIGRID (named, say, BIGRD1) in which the name of the GINDEX common block is changed to GINDX1.
(c) Make a version of the DEPTHS array (named DEPTH1) in which you change zero depth values indicating islands into small nonzero numbers in the 0.001 .... 0.01 m range.
(d) In the initialization part of the model, insert the statement CALL BIGRD1(DEPTH1) after CALL BIGRID(DEPTHS).

The solution produced by the above procedure, which amounts to covering islands with a thin sheet of water, should be sufficiently accurate for most display purposes. It is implemented in our model output processor MICOMPROC.

(10) Numerous diagnostic messages in the code are presently commented out by the letters CDIAG but can be activated as needed. Many of these messages refer to a specific grid point (ITEST,JTEST) which the user must define in a DATA statement in BLKDAT. Subroutine PRTIJ prints clusters of 5x5 grid variables centered on ITEST,JTEST in their proper spatial context. (PRTIJ may not work near the coast if the code is compiled with -ei.)

(11) The model equations are solved on a regular grid overlying a Mercator projection of the earth's surface; this is to assure isotropy in grid resolution. Provisions for rotating the poles 90 degrees, i.e., defining a true meridian as the "equator" of the map projection, are made. Subroutine NEWOLD/OLDNEW gives the location of (rotated) model grid points in relation to a lat/lon grid and vice versa. However, interpolation routines for translating fields of variables back and forth are the responsibility of the user.

(12) Versions 1.3 and up read 2-degree North Atlantic basin depths from an ASCII file named DEPTH.51x56 and 7 sets of monthly forcing fields. Forcing fields for version 2.3 and up are in ASCII files named FORCING.TAU_X, FORCING.TAU_Y, FORCING.WINDSPD, FORCING.RADFLX, FORCING.PRECIP, FORCING .Q_STAR, and FORCING.T_STAR. These form a complete set of boundary conditions that can be used to test-run the model. Initial conditions – based on zonally averaged Levitus climatology – are generated internally by routine POFLAT. (A batch submit file for machines running under Unix is given in DEMORUN_2.5. Results from a 10-day test run on a 32-bit workstation (DEC-5000) are in DEMOOUT_2.5.)

(13) During the upgrade of model version 1.3 to 2.0, a bug was introduced into the mixed-layer buoyancy calculation [BUOYFL(I,J)=...] which was not discovered until 17 October 1992. The mod file 1PT3_TO_2PT0.MODS was corrected on that day; this led to a set of altered source files for versions 2.0 - 2.2 which are marked with the suffix "BUGFIX".

(14) A FORTRAN-90 version of MICOM level 2.5 suitable for running on a Connection Machine (CM-5) is kept as a tar file in ftp subdirectory ANONYMOUS .BLECK.MASSIVE on nutmeg.rsmas.miami.edu . This particular version is based on 127 x 127 horizontal points with a mesh size of 0.9 deg. longitude. Forcing functions and a 127 x 127 point depth array are provided in the same subdirectory.

(15) Starting with version 2.4, the previously monolithic 2500-line main program has been split into approximately 10 subroutines. The fragmented code, which runs 10fragments more easily), requires the UNIX 'tar' utility for unpacking. The most recent version of the code is found in MICOM_2.5_TAR. It is being updated occasionally, mainly for cosmetic reasons. Users are advised to use the UNIX 'diff' utility to keep track of differences between their version and the latest version of MICOM.

(16) The major differences between versions 2.5 and 2.4 are as follows:

(a) mixed-layer dissipation is formulated according to Gaspar;
(b) the convective-adjustment algorithm in CONVEC.f entrains both momentum and mass into the mixed layer (it used to entrain only mass);
(c) thermal surface forcing is no longer part of MXLAYR.f but has been moved to a separate routine THERMF.f;
(d) time smoothing of u,v fields is done in a layer thickness-weighted fashion (analogous to the smoothing of mass field variables in TSADVC.f);
(e) the "physics" routines CONVEC,THERMF,MXLAYR are now called after the "dynamics" routines to remove inconsistencies in the definition and treatment of "old" and "new" fields.

(17) We are presently trying to improve the efficiency of the model by switching from leapfrog to forward-backward time differencing with Adams- Bashforth treatment of the inertial terms in the baroclinic momentum equations. Also planned is a generalization of the mixed-layer detrainment algorithm to allow downward salinity transfer within the fossil mixed layer in analogy to the presently occurring upward heat transfer. This is expected to speed up mixed-layer detrainment in polar regions.

(18) Setting horizontal array dimensions IDM,JDM to powers of 2 may cause slowdowns on some computers due to cache or memory bank conflicts. Since the scope of do-loops in the model is defined in terms of the variables II, JJ, model performance can be optimized without changing the physical basin configuration by altering IDM and/or JDM but not II,JJ. (Needless to say, IDM,JDM may not be smaller than II,JJ respectively.)

These NOTES by :
Rainer Bleck (rbleck@rsmas.miami.edu)
13 April 1994

References

BARAILLE R., FILATOFF N., 1995 : Modèle Shallow-water Muticouches Isopcynal de Miami. *Rapport d'Etude* CMO/RE No 003/95

BLECK R., 1978 : Finite Difference Equations in General Vertical Coordinates. *Contrib. Atmos. Phys*, **51**, 360-372

BLECK R., BOUDRA D.B., 1986 : Wind-driven Spin-up in Eddy-resolving Ocean Models Formulated in Isopycnic and Isobaric Coordinates. *J. Geophys. Res.*, **91**(C), 7611-7621

BLECK R., HANSON H.H., HU D., KRAUS E.B., 1989 : Mixed Layer-Thermocline Interaction in a 3D Isopycnic Coordinate Model. *J. Phys. Oceanogr.*, **19**, 1417-1439

BLECK R., SMITH L.T., 1990 : A Wind-driven Isopycnic Coordinate Model of the North and Equatorial Atlantic Ocean. I- Model Development and Supporting Experiments. *J. Geophys. Res.*, **95**(C), 3273-3285

BLECK R., ROOTH C., HU D., SMITH L.T., 1992 : Salinity-driven Thermocline Transients in a Wind- and Thermohaline-forced Isopycnic Coordinate Model of the North Atlantic. *J. Phys. Oceanogr.*, **22**, 1486-1505

CUSHMAN-ROISIN B., 1994 : *Introduction to Geophysical Fluid Dynamics.* **Prentice Hall**, New-Jersey.

FRIEDRICH H., LEVITUS S., 1972 : An Approximation to the Equation of State for Sea Water, Suitable for Numerical Ocean Models. *J. Phys. Oceanogr.*, **2**, 514-517

GASPAR P., 1988 : Modelling the Seasonal Cycle of the Upper Ocean. *J. Phys. Oceanogr.*, **18**, 161-180

HU D., 1991 : A Joint Mixed Layer/Isopycnic Coordinate Numerical Model of Wind- and Thermohaline- driven Ocean General Circulation with Model Sensitive Study. *Ph.D.*, University of Miami, Florida, 220 pp

KRAUS E.B., TURNER J.S, 1967 : A One-dimensional Model of the Seasonal Thermocline. Part II : The General Theory and its Consequences. *Tellus*, **19**, 98-105

NIILER P.P., KRAUS E.B., 1977 : A One-dimensional Model of the Upper Ocean. In *Modelling and Prediction of the Upper Layers of the Ocean.* **Pergamon Press.**

SMOLARKIEWICZ P.K., 1984 : A Fully Multidimensional Positive Definite Advection Transport with Small Implicit Diffusion. *J. Comput. Phys.*, **54**, 325-362

SMOLARKIEWICZ P.K., CLARK T.L., 1986 : The Multidimensional Positive Definite Advection Transport Algorithm : Further Development and Applications. *J. Comput. Phys.*, **67**, 396-438

SMOLARKIEWICZ P.K., GRABOWSKI W.W., 1990 : The Multidimensional Positive Definite Advection Transport Algorithm : Non-oscillatory Options. *J. Comput. Phys.*, **86**, 355-375

SUN S., BLECK R., CHASSIGNET E.P., 1993 : Layer Outcropping in Numerical Models of Stratified Flows. *J. Phys. Oceanogr.*, **23**, 1877-1884

ZALESAK, S., 1979 : Fully Multidimensional Flux-corrected Transport Algorithms for Fluids. *J. Comput. Phys., 31*, 335–362.